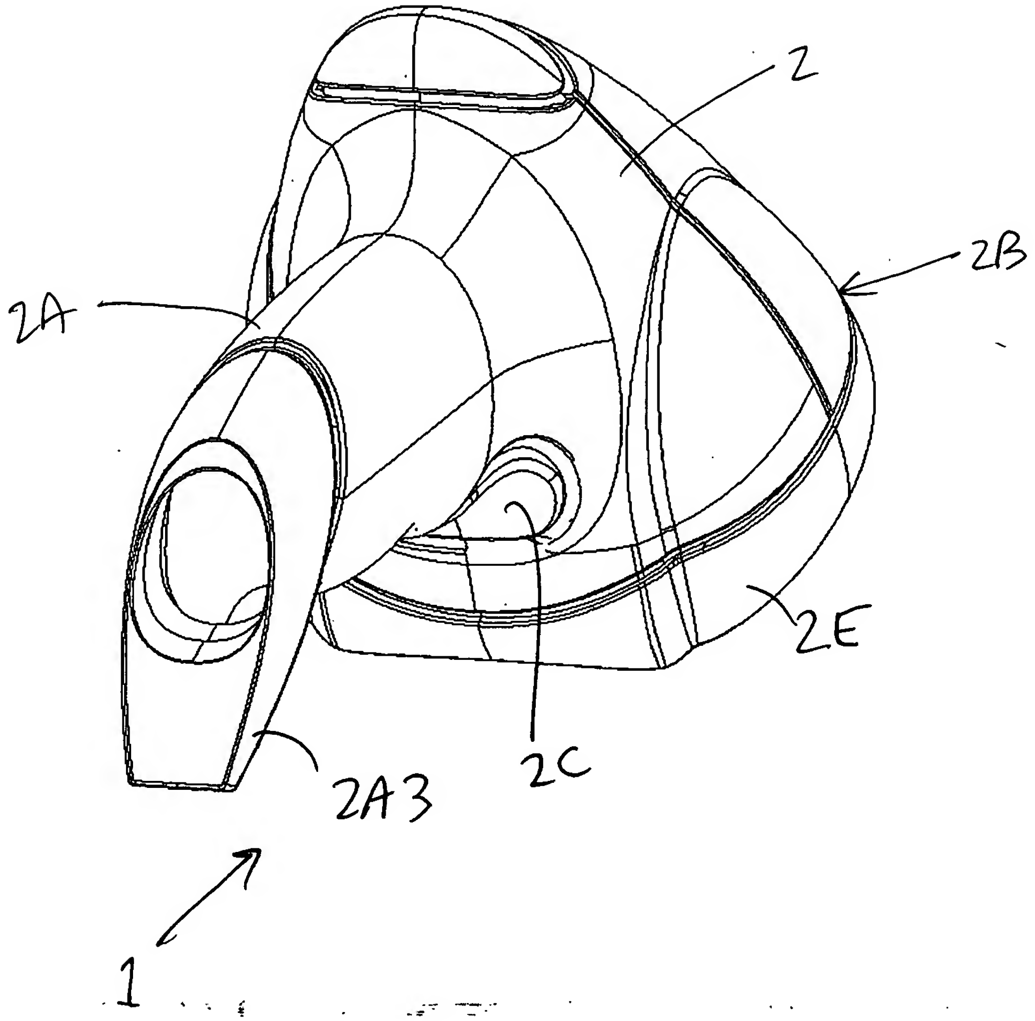


FIG. 1A

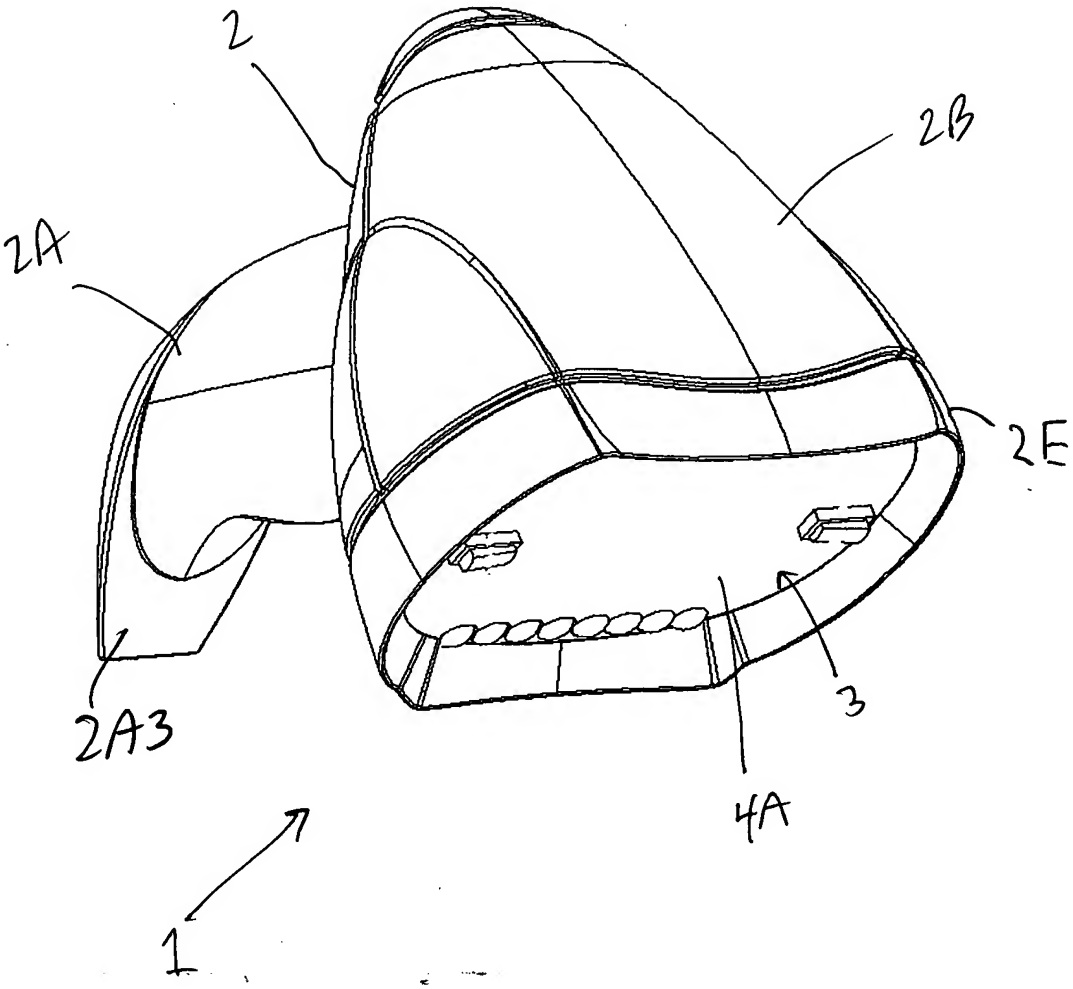


FIG. 1B

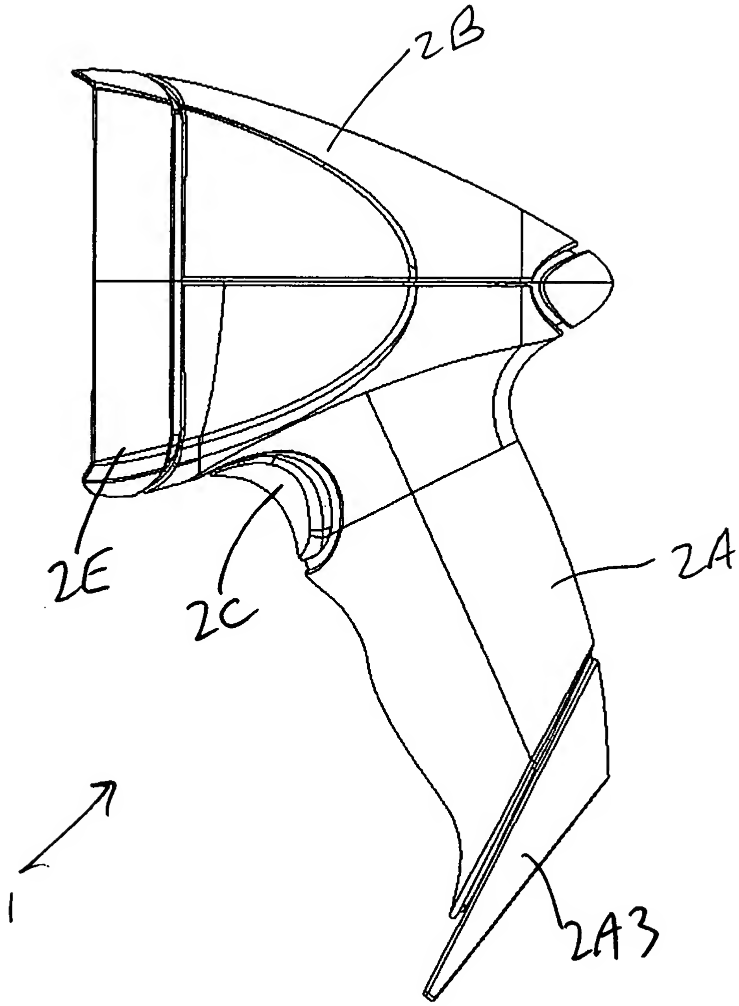


FIG. 1C

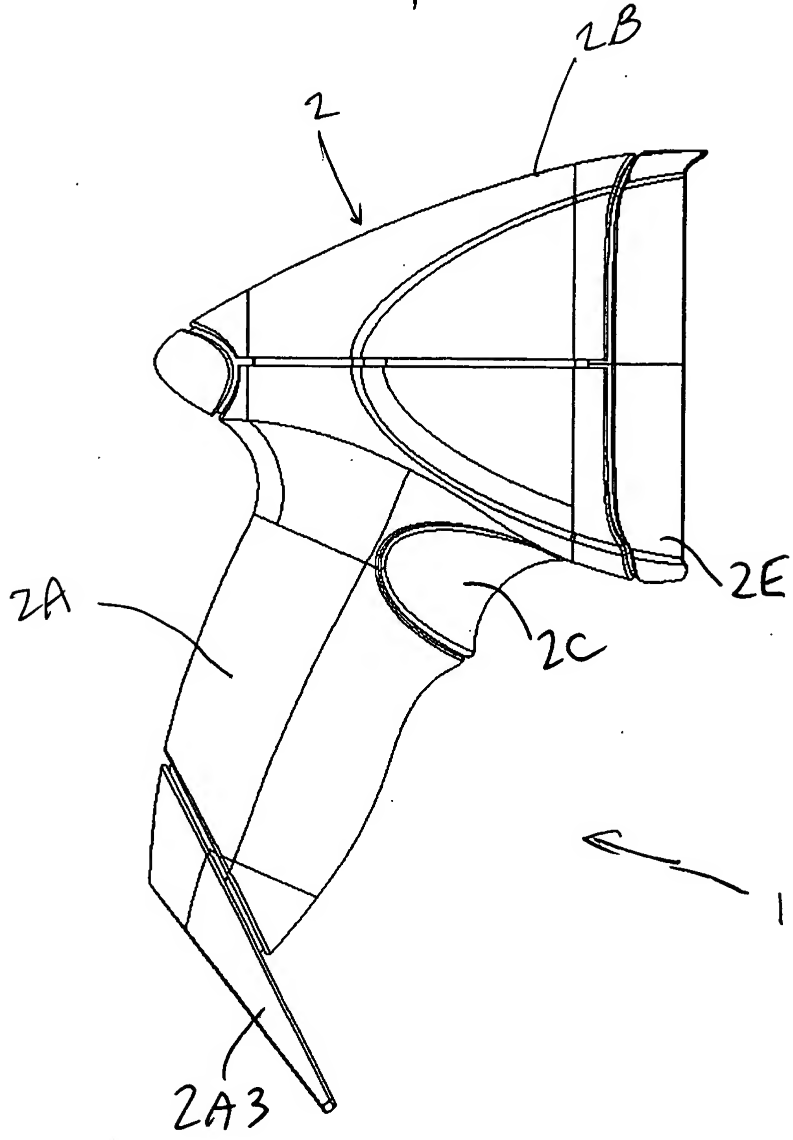


FIG. 1D

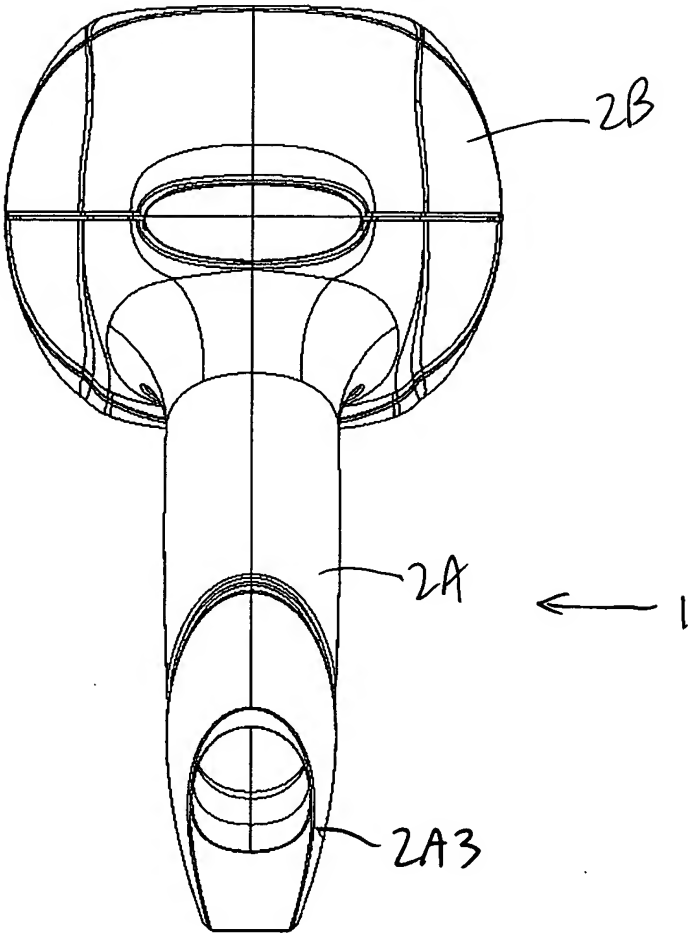


FIG. 1E

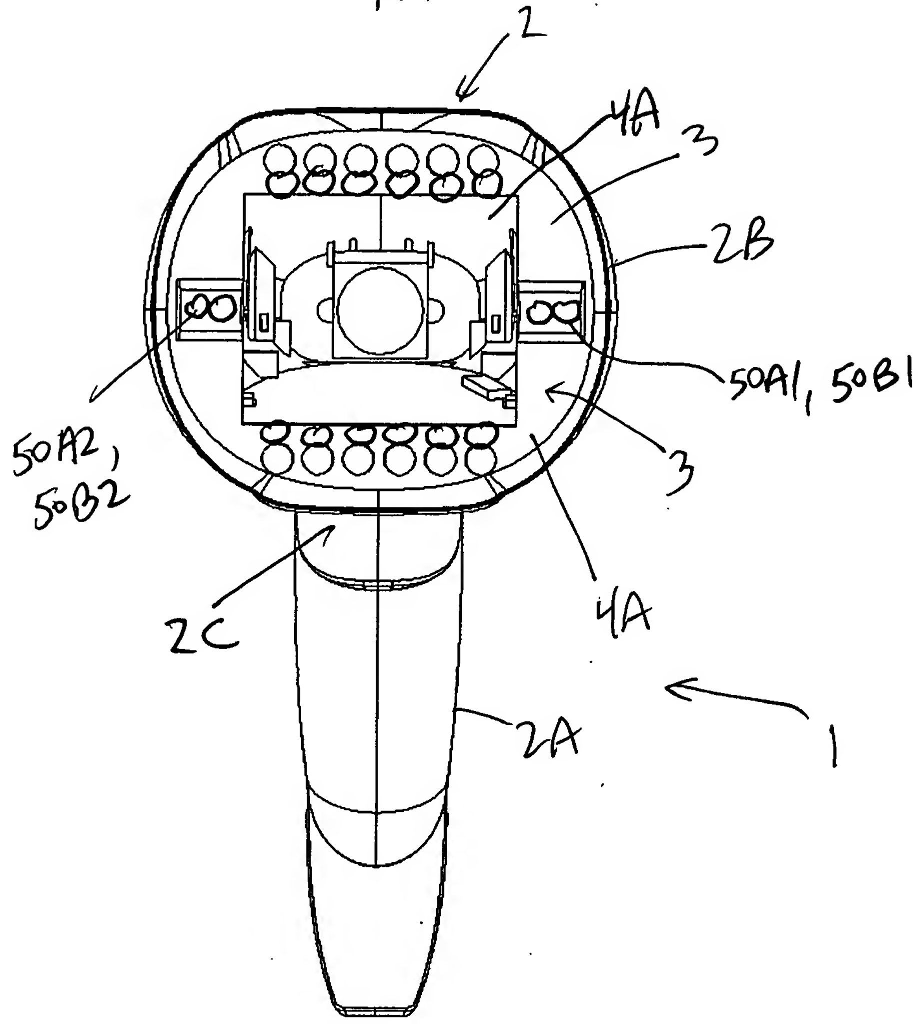


FIG. 1F

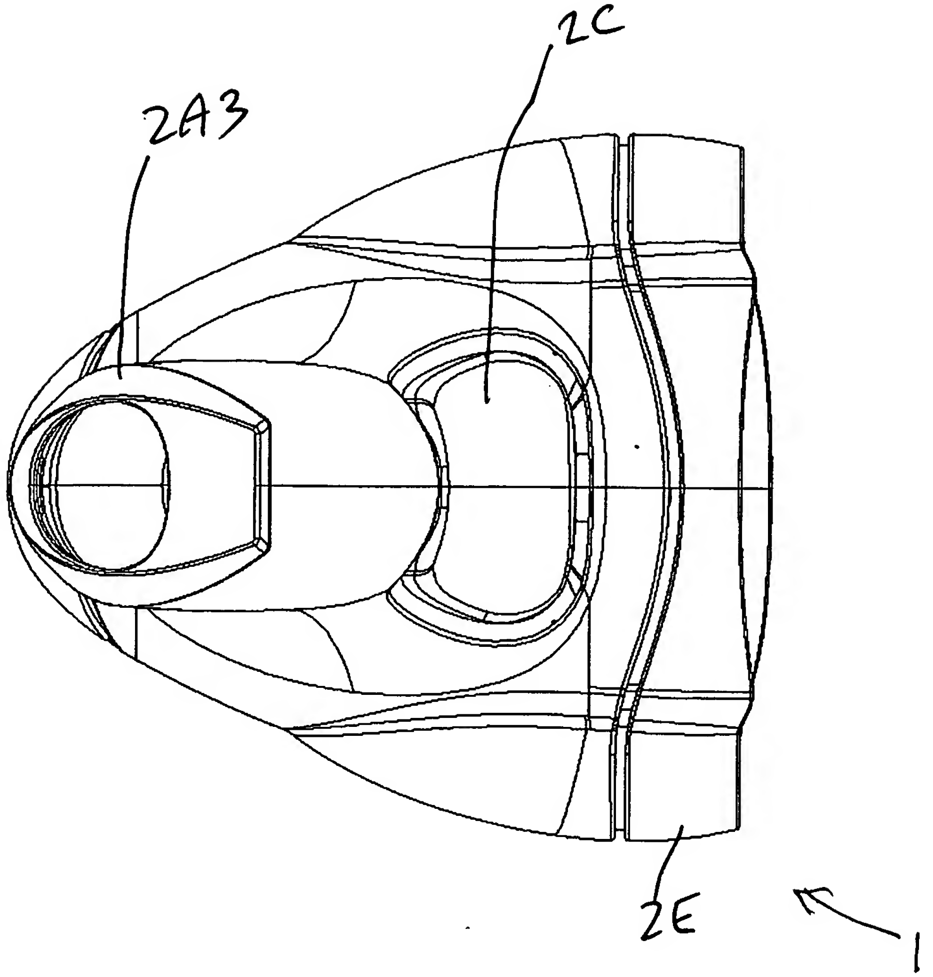


FIG. 1G

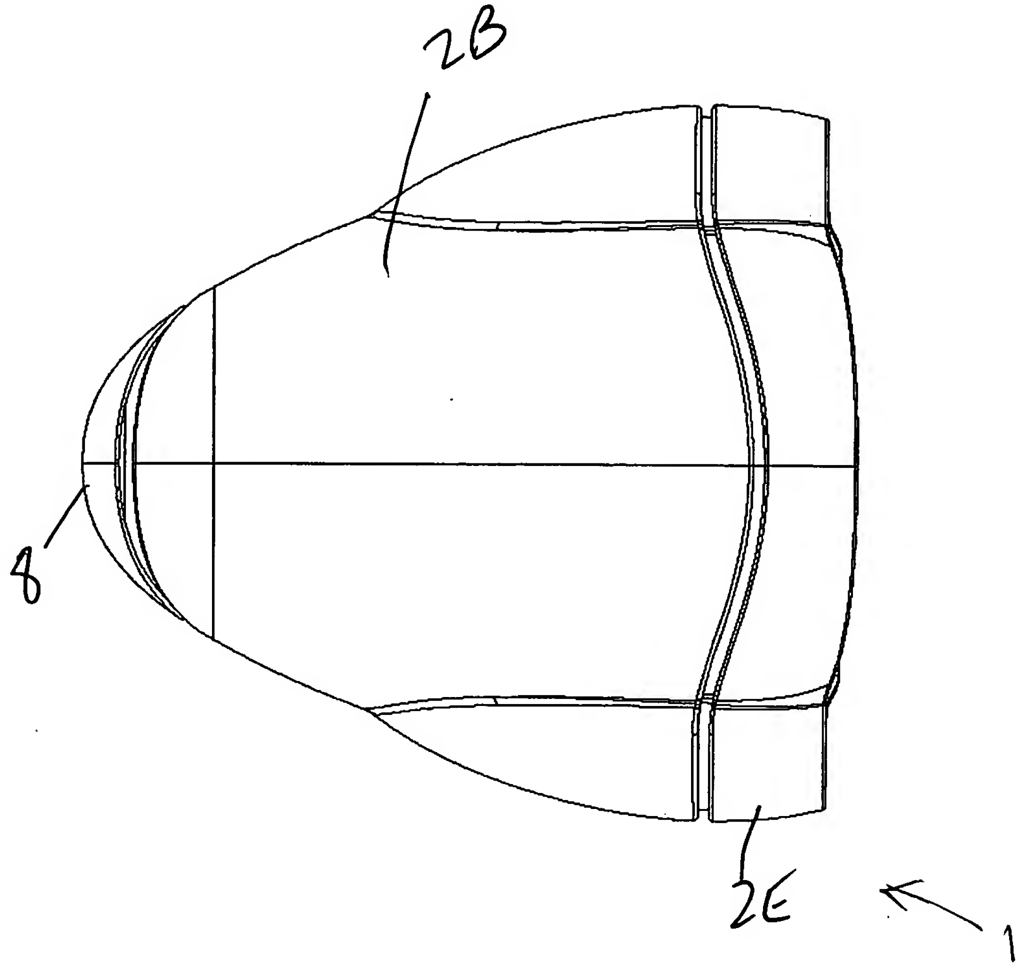


FIG. 1H

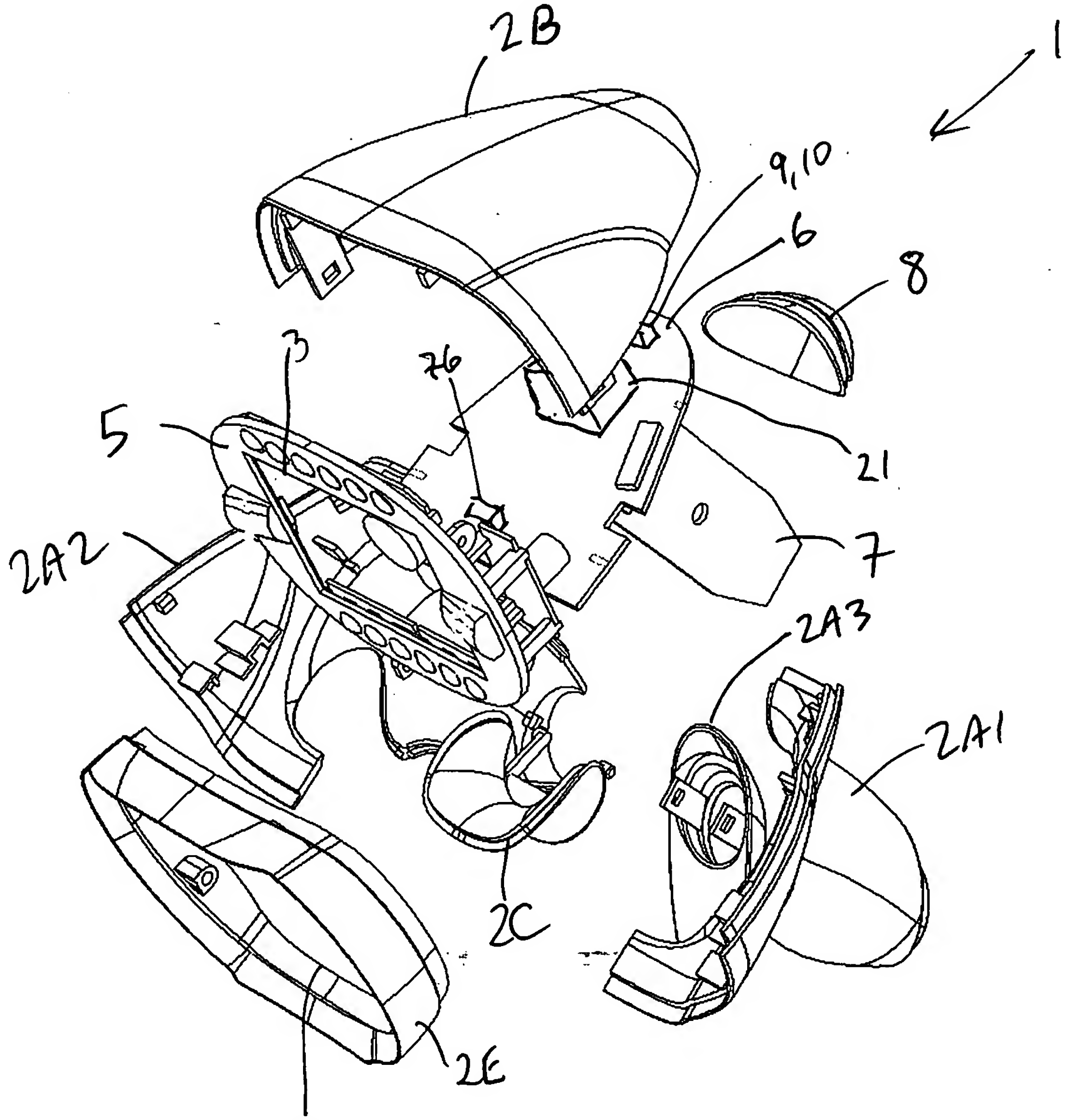


FIG. 1I

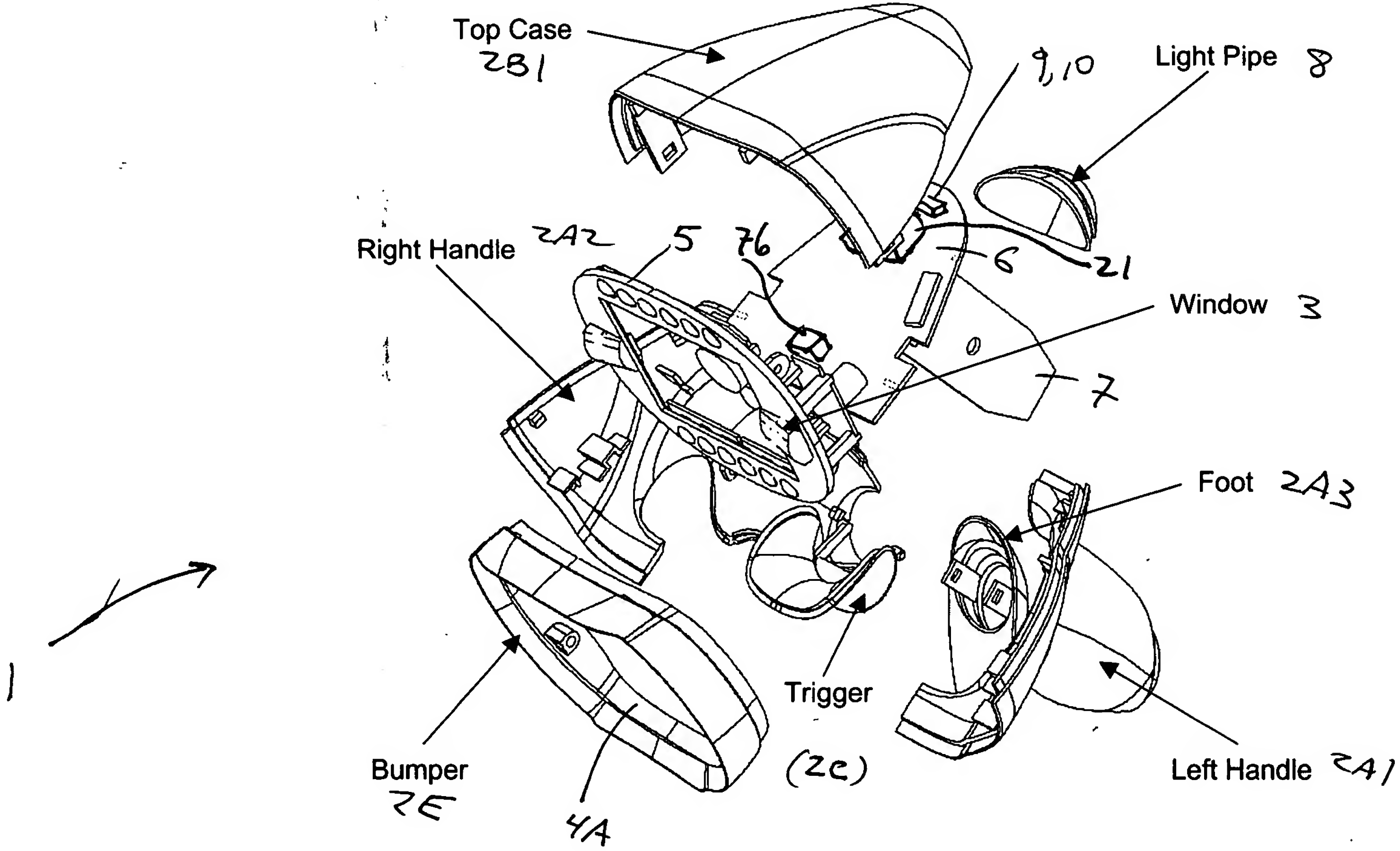

Top Case
2B1
9,10
Light Pipe 8
Right Handle 2A2
5
76
6
21
Window 3
7
Foot 2A3
1
Trigger
(2C)
Bumper
2E
4A
Left Handle 2A1

FIG. 1J

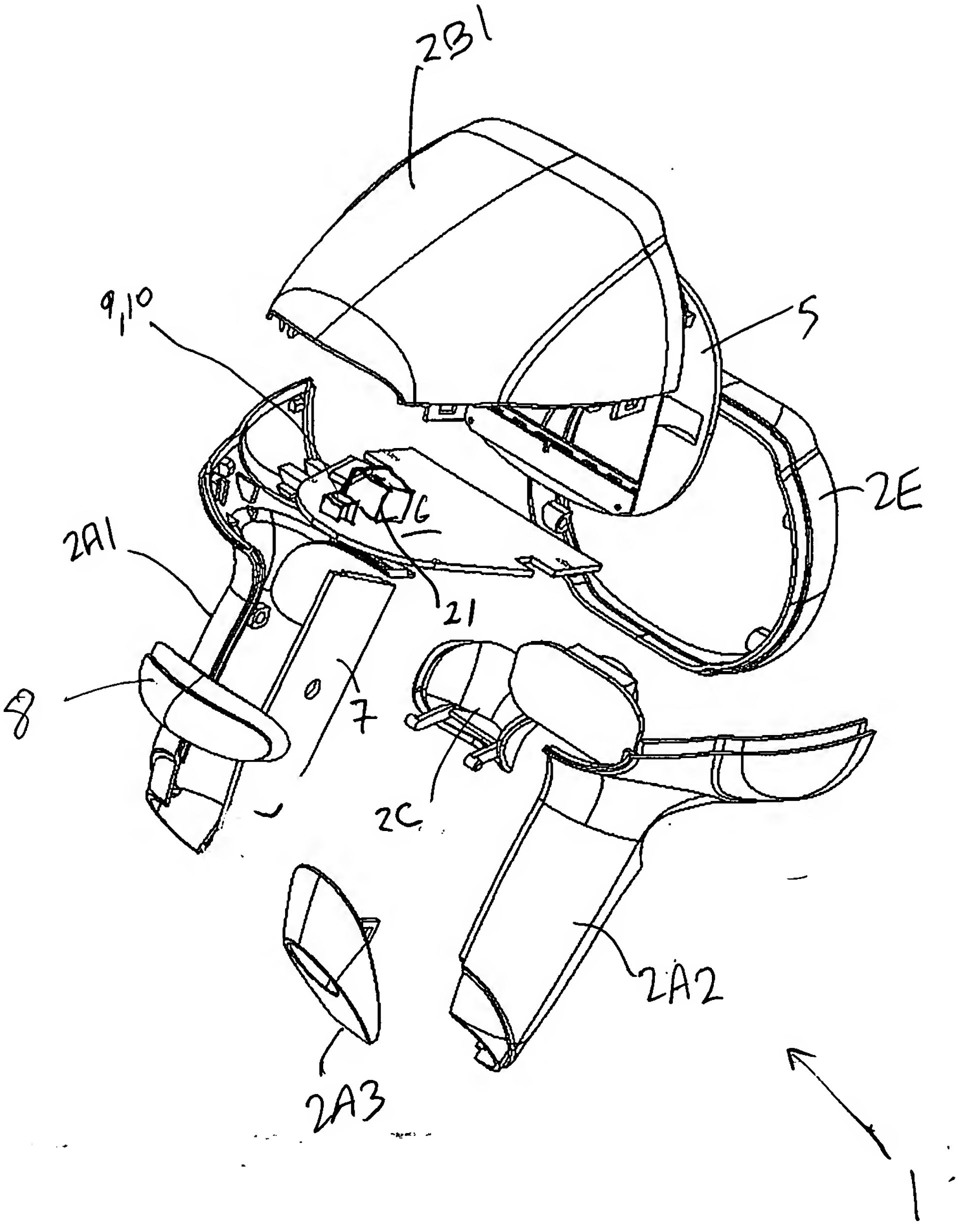


FIG. 1K

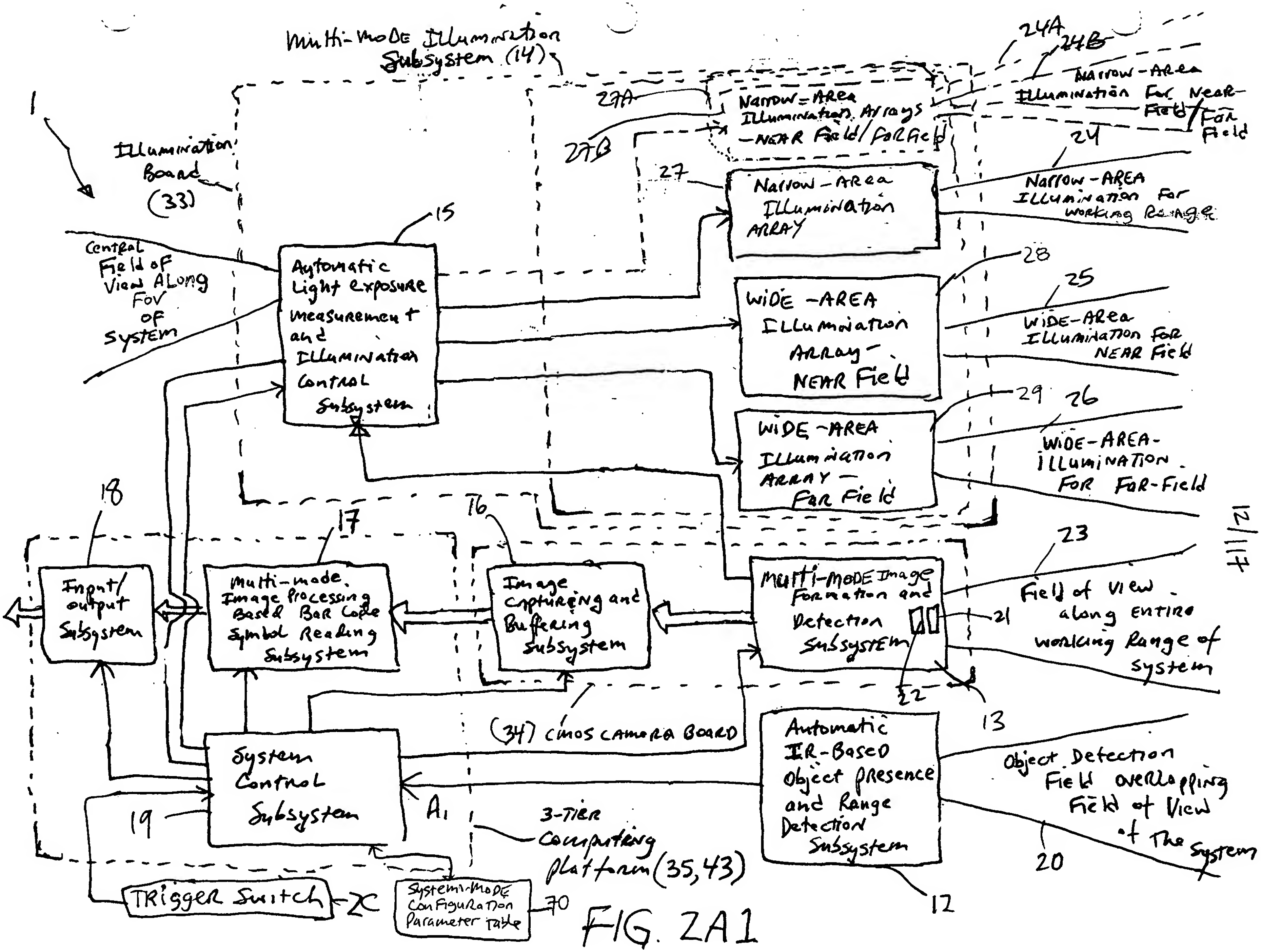

Multi-mode Illumination Subsystem (14)
1
Illumination Board (33)
Central Field of View Along FOV of System
15
Automatic Light exposure measurement and Illumination Control Subsystem
27A
27B
27
24A
24B
Narrow-Area Illumination Arrays -NEAR Field/FarField
Narrow-Area Illumination For Near-Field/Far Field
24
Narrow-Area Illumination Array
Narrow-AREA Illumination For working Range
28
25
WIDE-AREA Illumination Array-NEAR Field
WIDE-AREA Illumination For NEAR Field
29
26
WIDE-AREA Illumination ARRAY-Far Field
WIDE-AREA-ILLUMINATION FOR FAR-Field
18
Input/output Subsystem
17
Multi-mode Image Processing Based Bar Code Symbol Reading Subsystem
16
Image Capturing and Buffering Subsystem
Multi-MODE Image Formation and Detection Subsystem
23
Field of View along entire working Range of system
21
22
13
(34) CMOS CAMERA BOARD
System Control Subsystem
A1
19
3-TIER Computing platform (35,43)
Automatic IR-Based Object presence and Range Detection Subsystem
Object Detection Field overlapping Field of View of The System
20
12
TRIGGER Switch
2C
System-Mode Configuration Parameter Table
70

FIG. 2A1

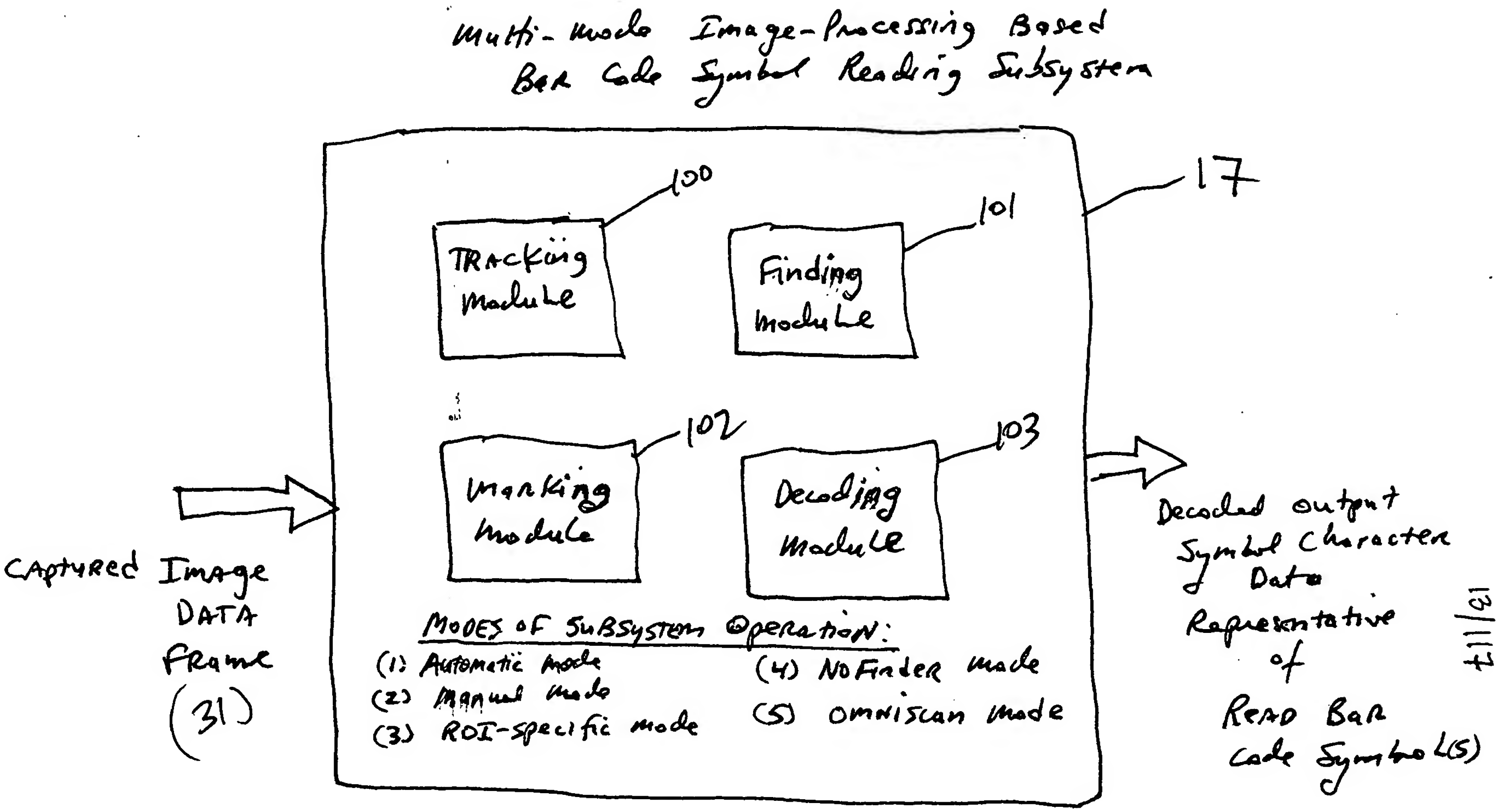

Multi-mode Image-Processing Based
Bar Code Symbol Reading Subsystem
17
100
Tracking Module
101
Finding Module
102
Marking Module
103
Decoding Module
MODES OF SUBSYSTEM Operation:
(1) Automatic mode
(2) Manual mode
(3) ROI-specific mode
(4) NoFinder mode
(5) Omniscan mode
Captured Image DATA Frame (31)
Decoded output Symbol Character Data Representative of Read Bar Code Symbol(s)

FIG. 2A2

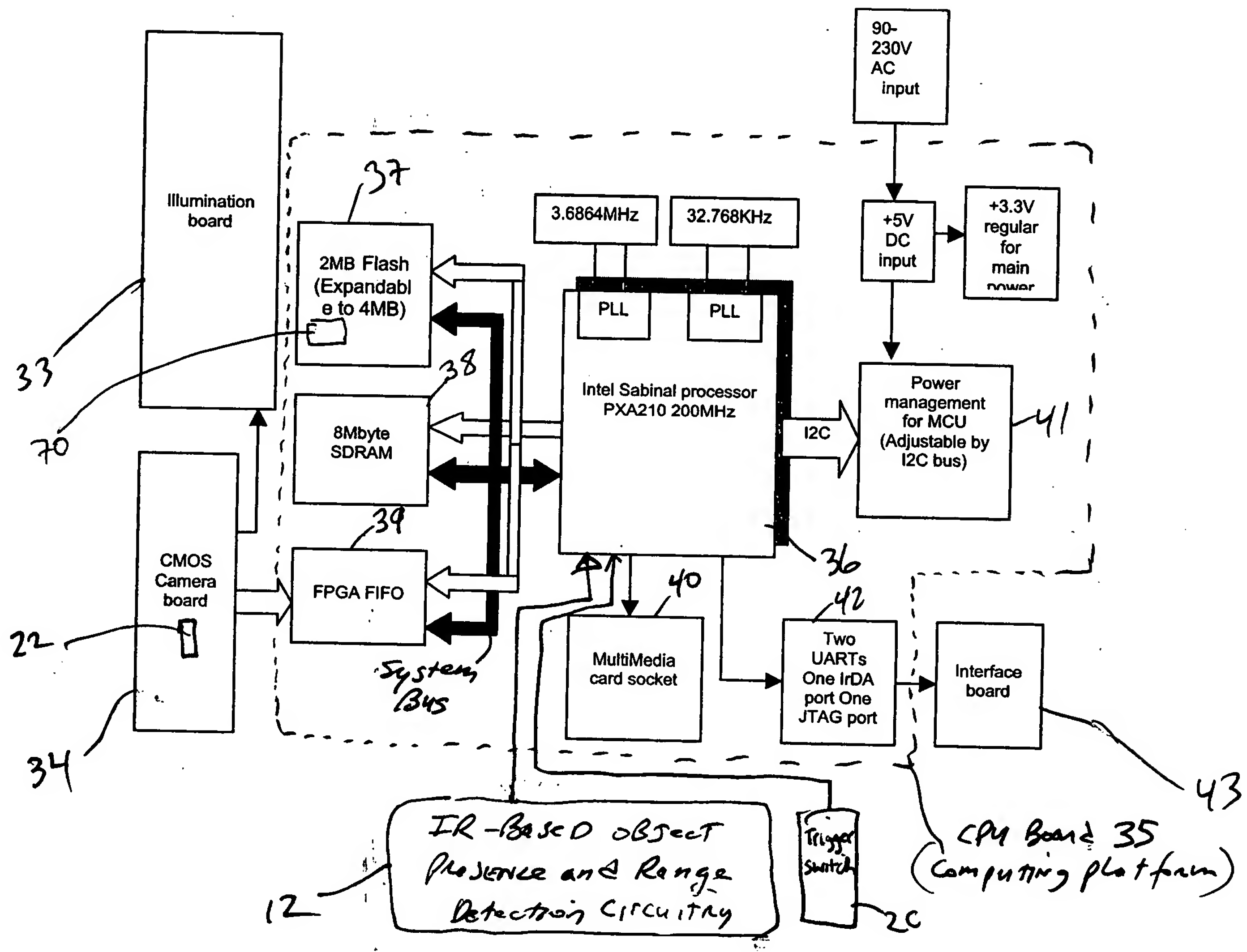

90-
230V
AC
input
+5V
DC
input
+3.3V
regular
for
main
power
Power
management
for MCU
(Adjustable by
I2C bus)
41
I2C
3.6864MHz
32.768KHz
PLL
PLL
Intel Sabinal processor
PXA210 200MHz
36
Illumination
board
33
37
2MB Flash
(Expandabl
e to 4MB)
70
38
8Mbyte
SDRAM
39
FPGA FIFO
System
Bus
CMOS
Camera
board
22
34
40
MultiMedia
card socket
42
Two
UARTs
One IrDA
port One
JTAG port
Interface
board
43
CPU Board 35
(Computing platform)
IR-Based object
Presence and Range
Detection Circuitry
12
Trigger
switch
2C

FIG. 2B

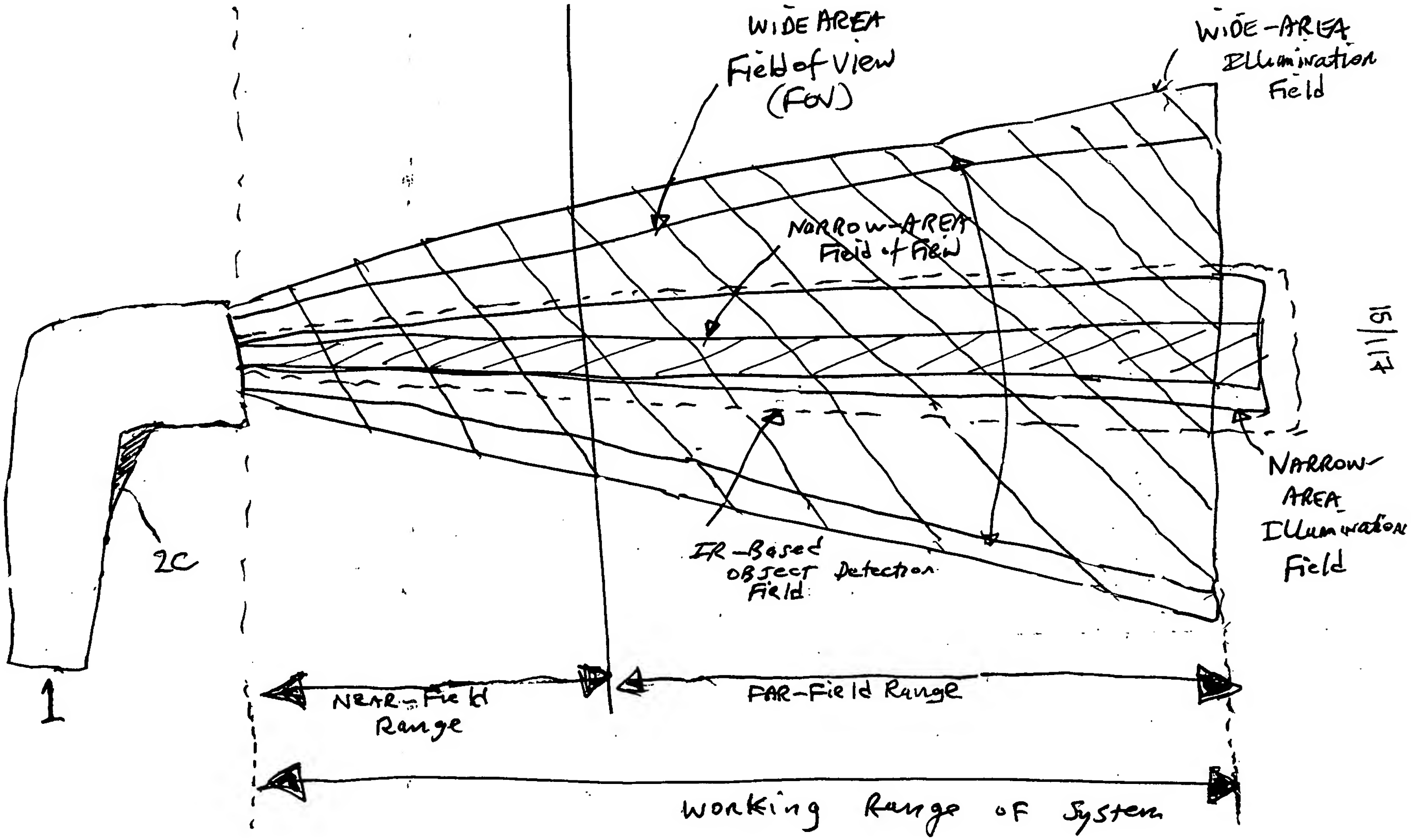

WIDE AREA Field of View (FOV)
WIDE-AREA Illumination Field
NARROW-AREA Field of View
NARROW-AREA Illumination Field
IR-Based object Detection Field
2C
1
NEAR-Field Range
FAR-Field Range
Working Range of System

FIG. 3A

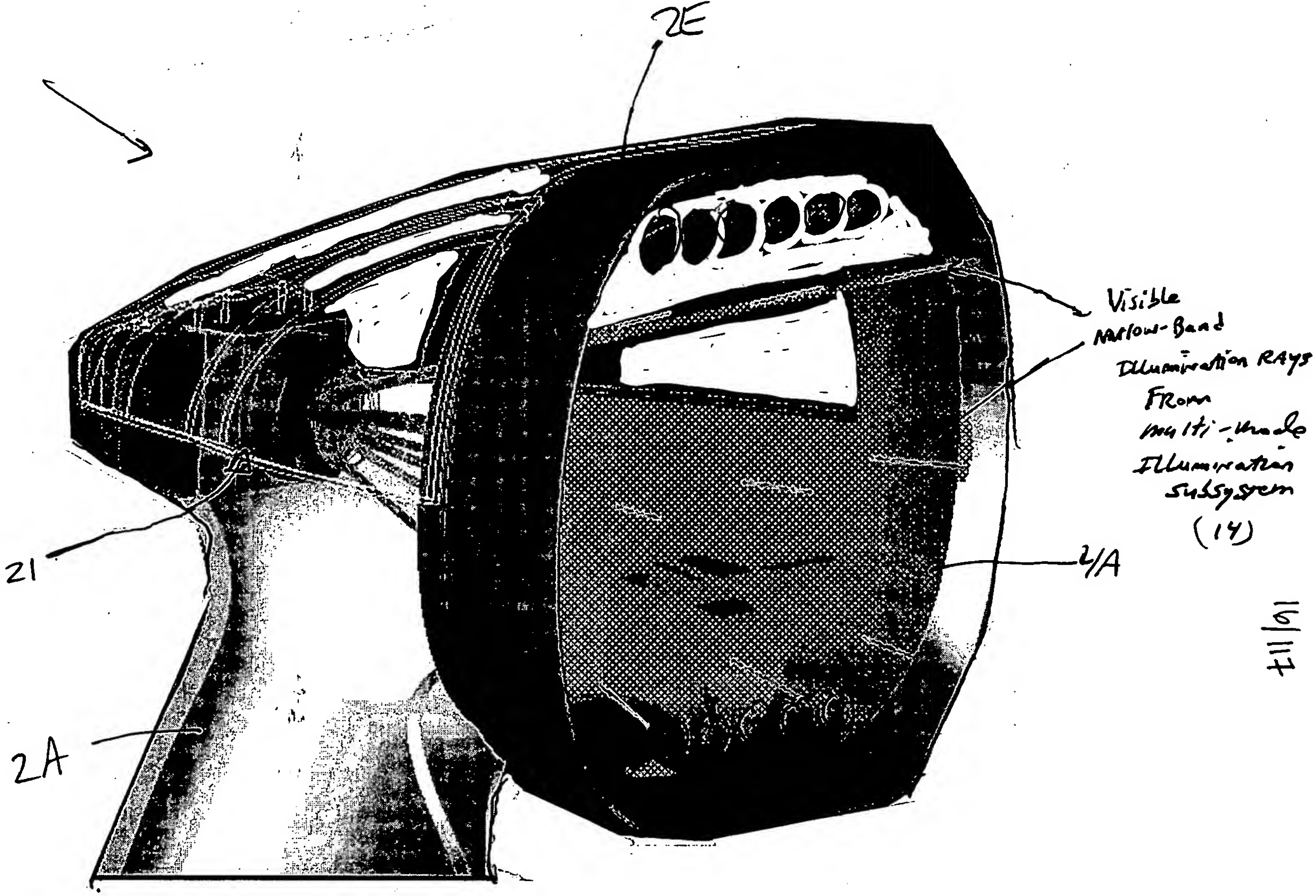
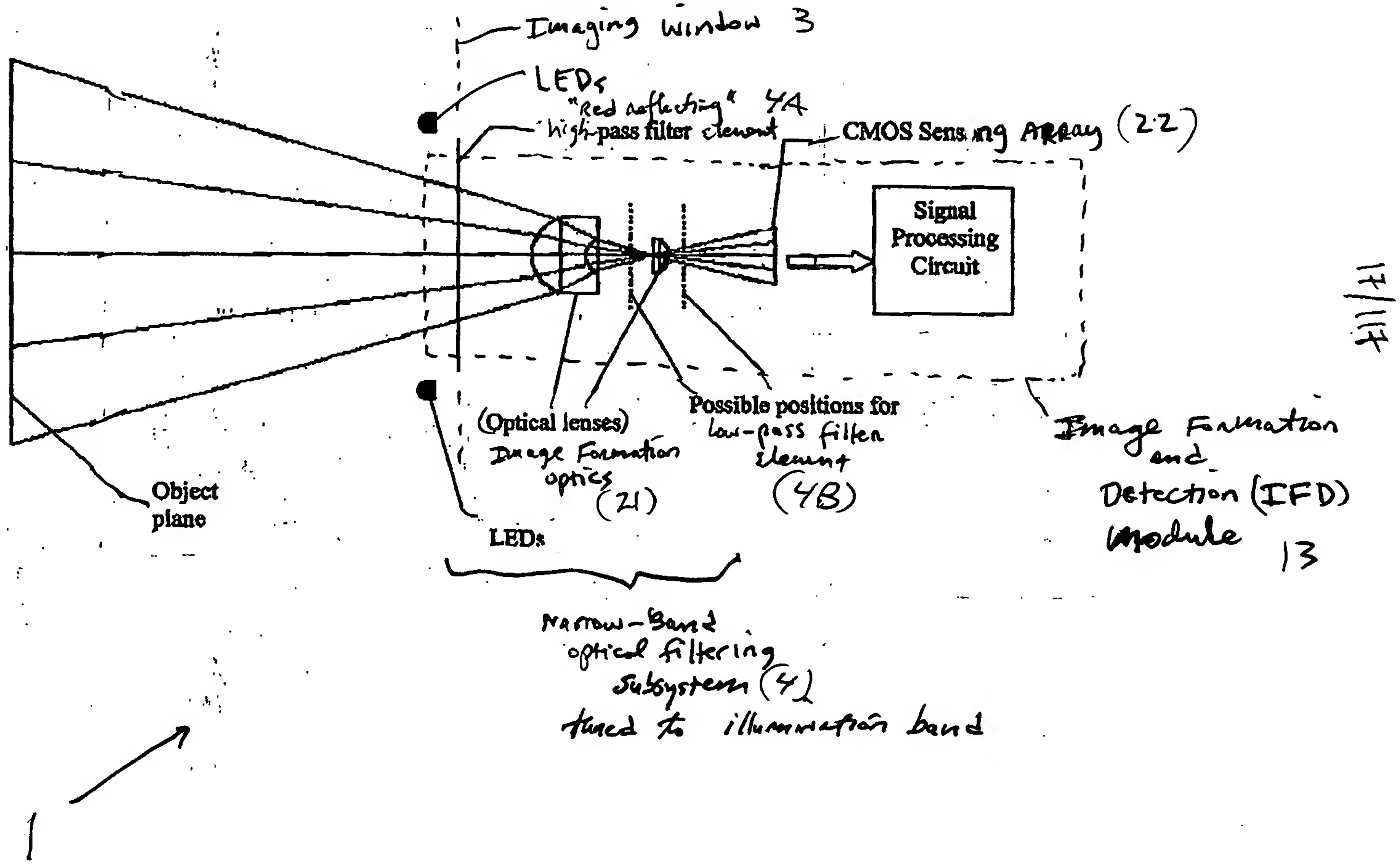

2E
Visible
Narrow-Band
Illumination Rays
From
multi-mode
Illumination
Subsystem
(14)
4A
21
2A

FIG. 3B



Imaging window 3
LEDs
"Red reflecting" 4A
high-pass filter element
CMOS Sensing Array (2.2)
Signal Processing Circuit
(Optical lenses)
Image Formation optics (21)
Possible positions for low-pass filter element (4B)
Image Formation and Detection (IFD) Module 13
Object plane
LEDs
Narrow-Band optical filtering subsystem (4)
tuned to illumination band
1

FIG. 3C

- 45° FOV ✓
- As few elements as possible ✓
  - Previous designs had 4 or 5
- As small as possible ✓
  - Max diameter = 12 mm
- All spherical surfaces ✓
- Common glasses ✓
  - LAK2 (≈ LaK9)
  - ZF10 (= SF8)
  - LAF2 (≈ LaF3)

Image Formation optics 21

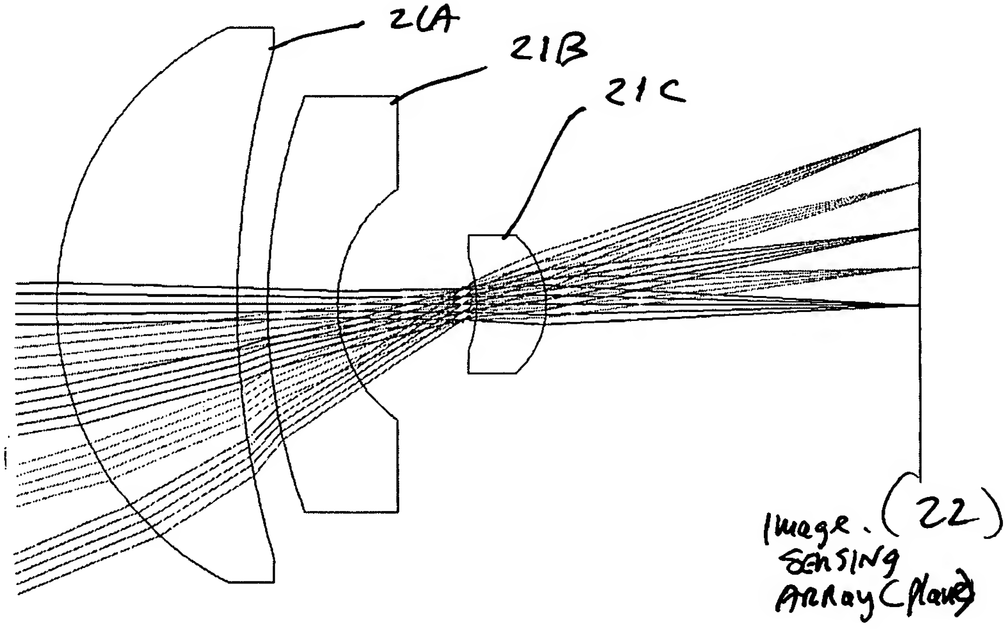


FIG. 3D

Image Formation optics assembly (21)

- Barrel hold lens elements
- Base holds sensor
- Barrel slides in base to focus

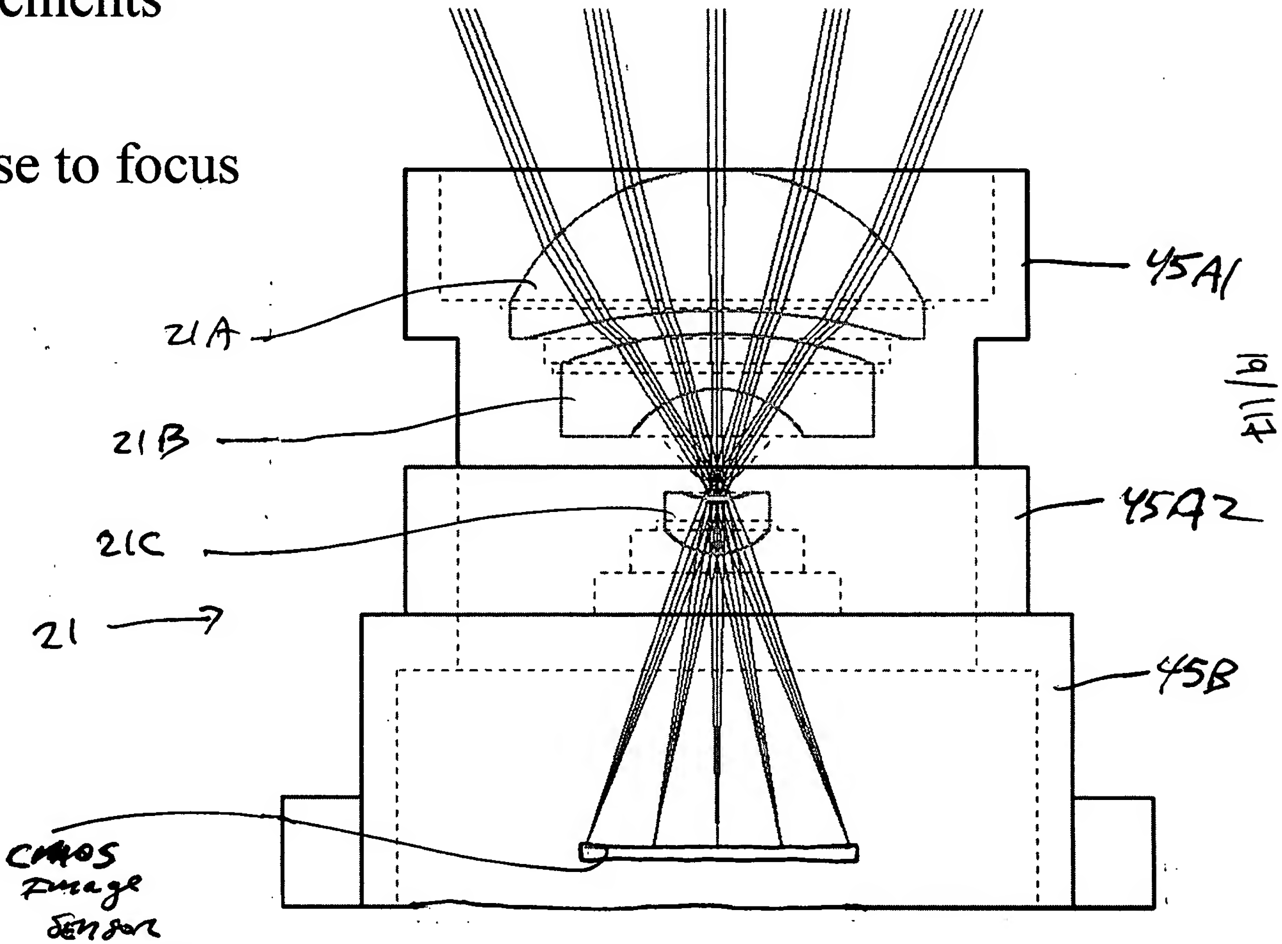


FIG.3E

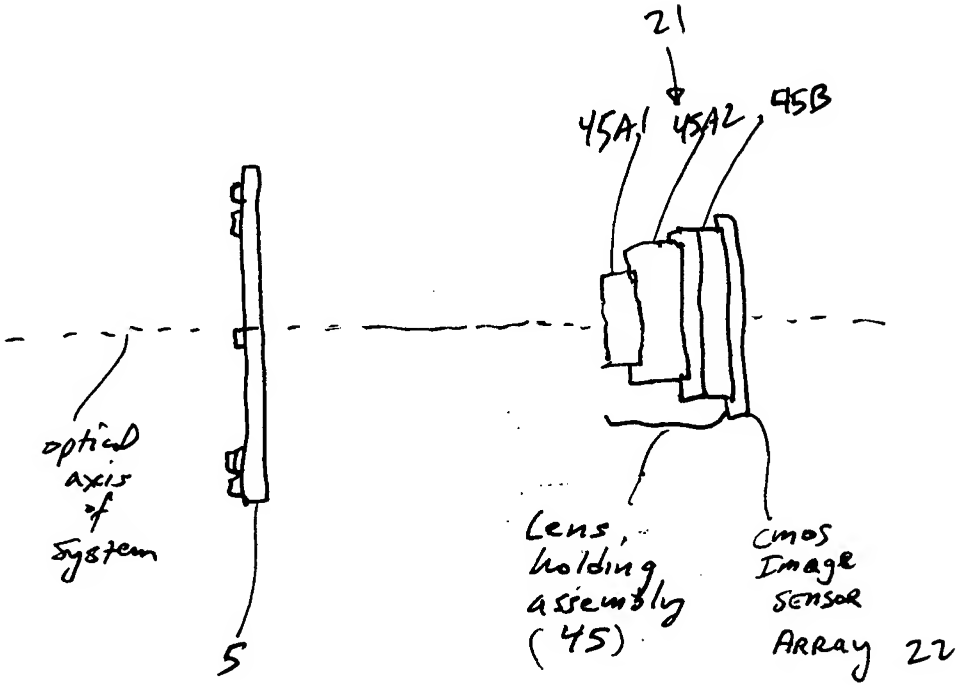


FIG. 3F1

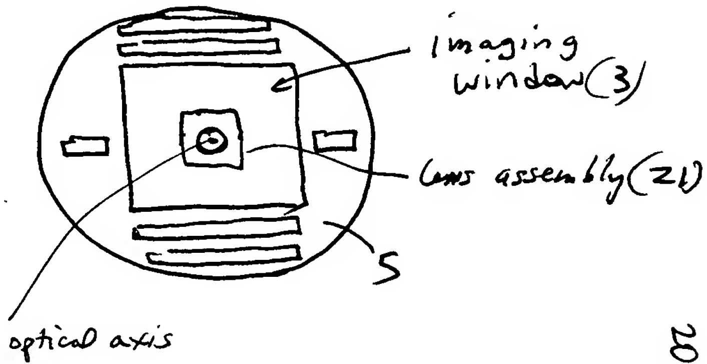
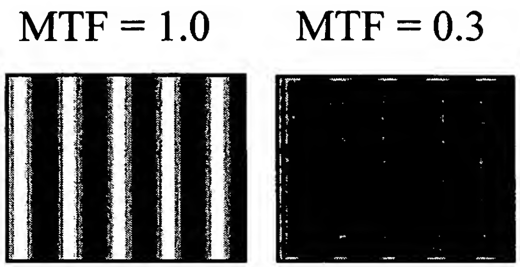


FIG. 3F2

DOF Determination *of Image Formation optics*

- At each distance, find frequency where MTF drops to 0.3
  - Rule of thumb for bar code decoding
  - Depends on code, speed, etc, etc - must test



- BUT: limited by sampling requirement
  - Software needs ~1.6 pixels on narrow code element
  - Limits decode ability regardless of optics
  - Exact value is rule of thumb and flexible (1.4 – 1.6)

FIG. 3G

Depth of Field

- Face to 8" for 13.5 mil ✓
- Optics resolve 4 mil somewhere ✓
- Decodes 5 mil somewhere ✓
- No moving elements ✓

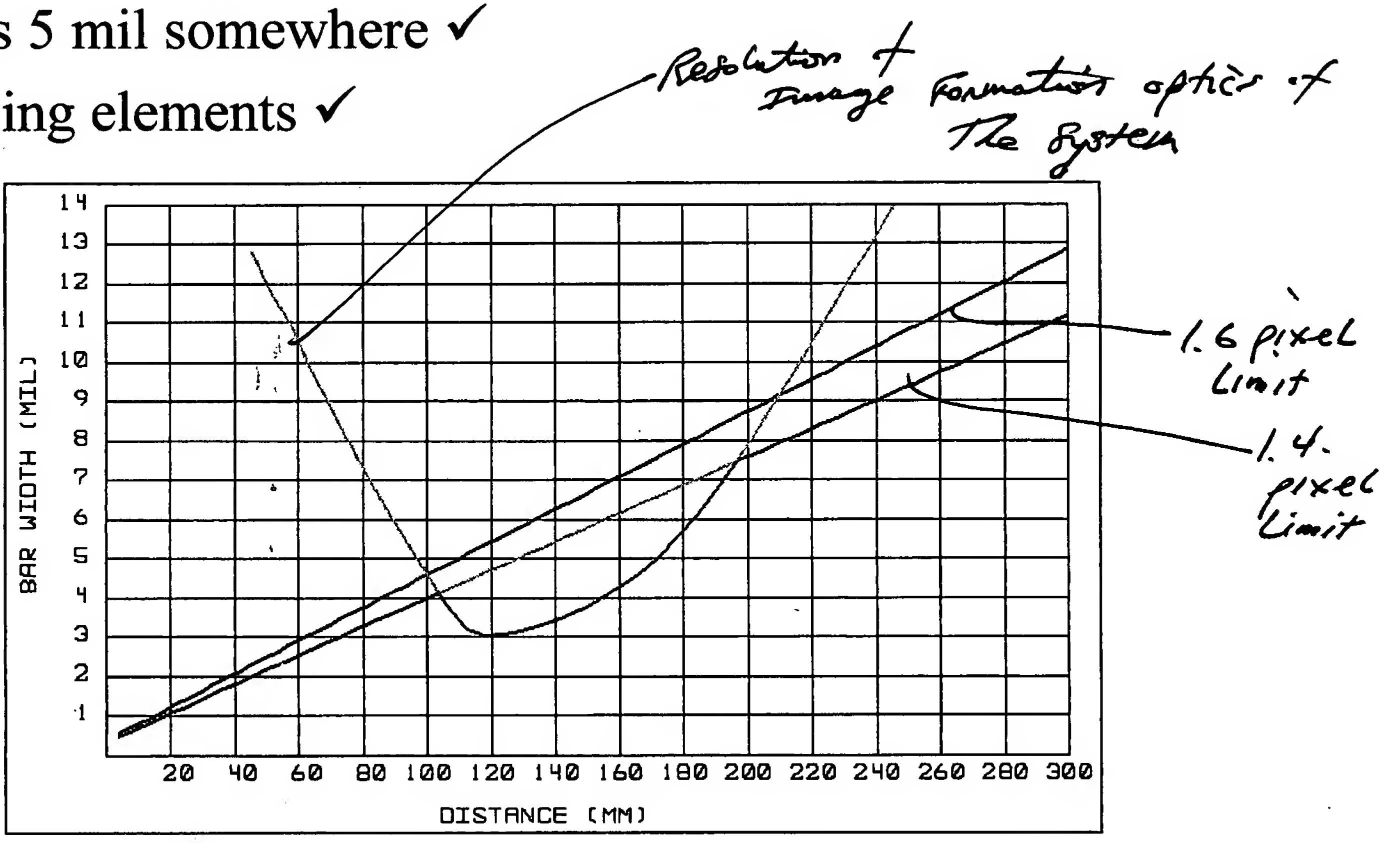


FIG. 4A

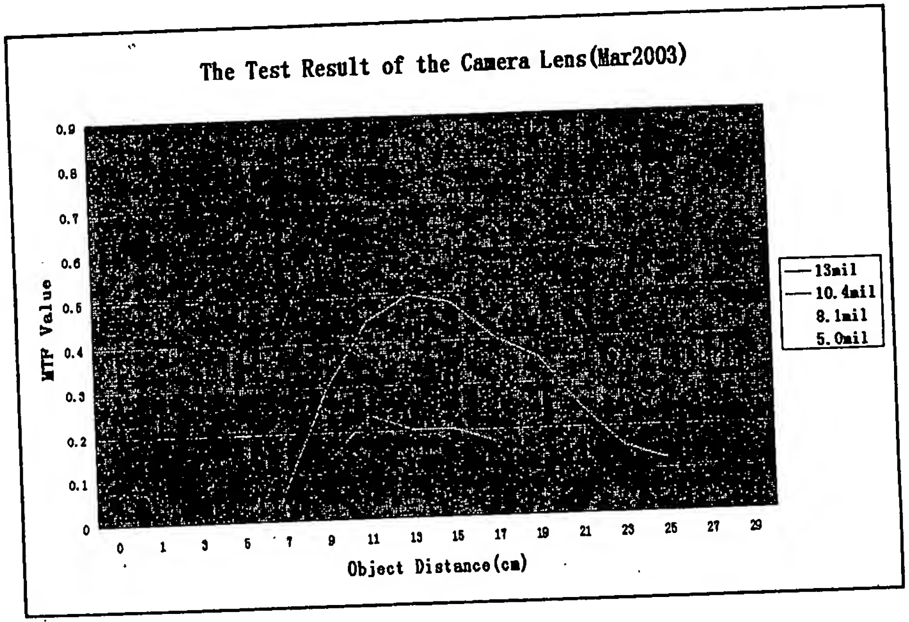

The Test Result of the Camera Lens(Mar2003)
MTF Value
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
1
3
5
7
9
11
13
15
17
19
21
23
25
27
29
Object Distance(cm)
13mil
10.4mil
8.1mil
5.0mil

FIG 4B

Depth of Field

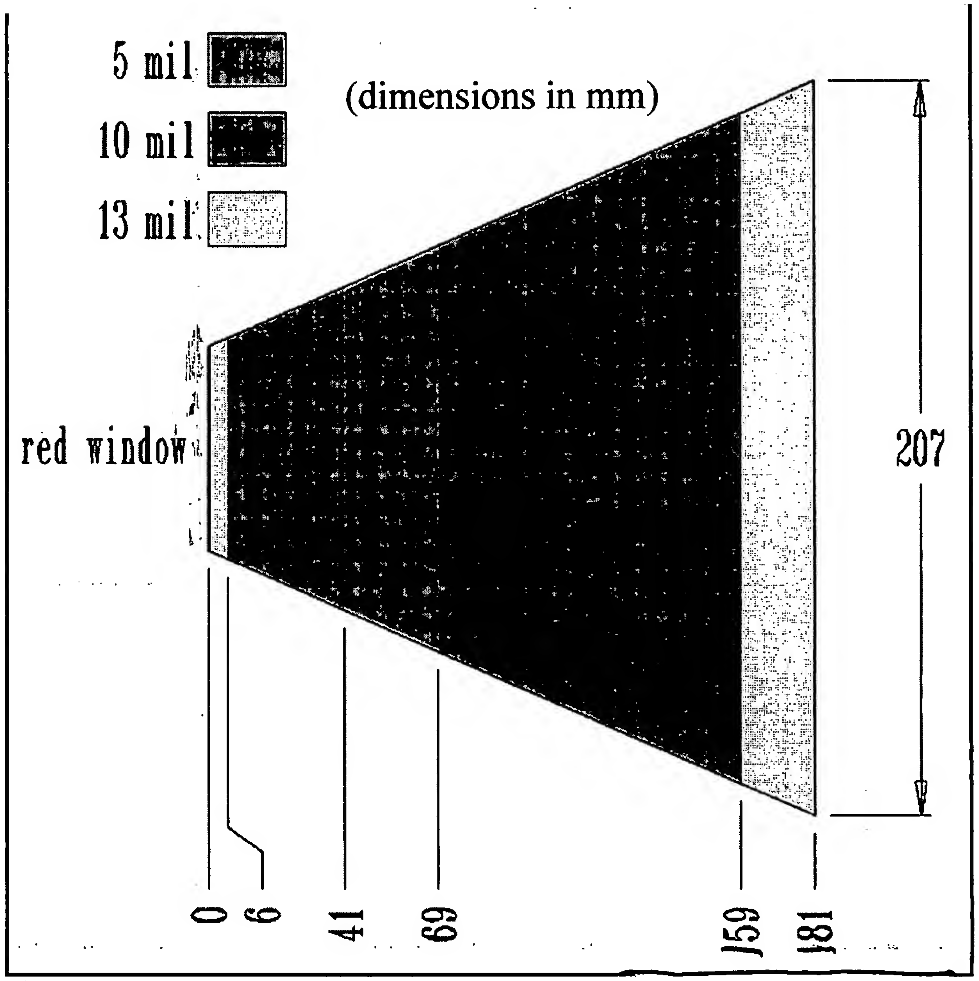
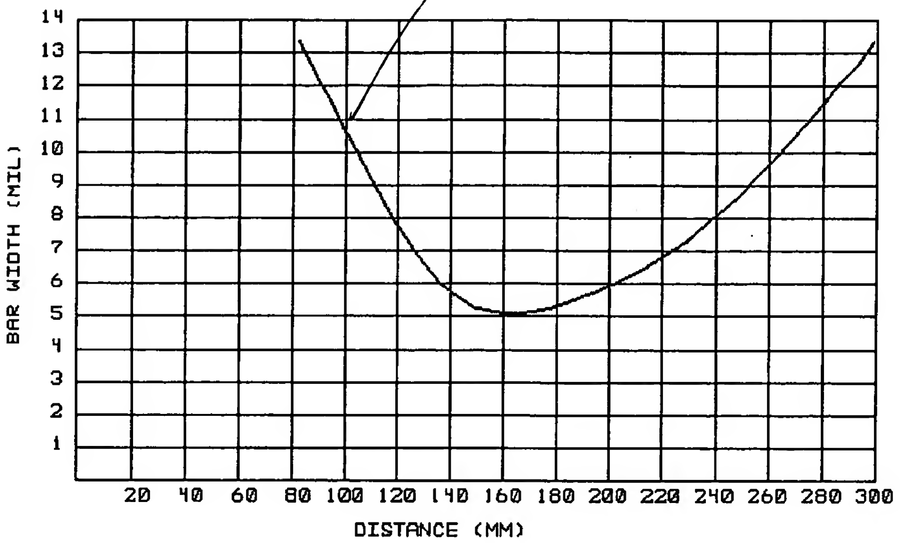


FIG. 4C



- Resolution of image formation optics

FIG. 4D

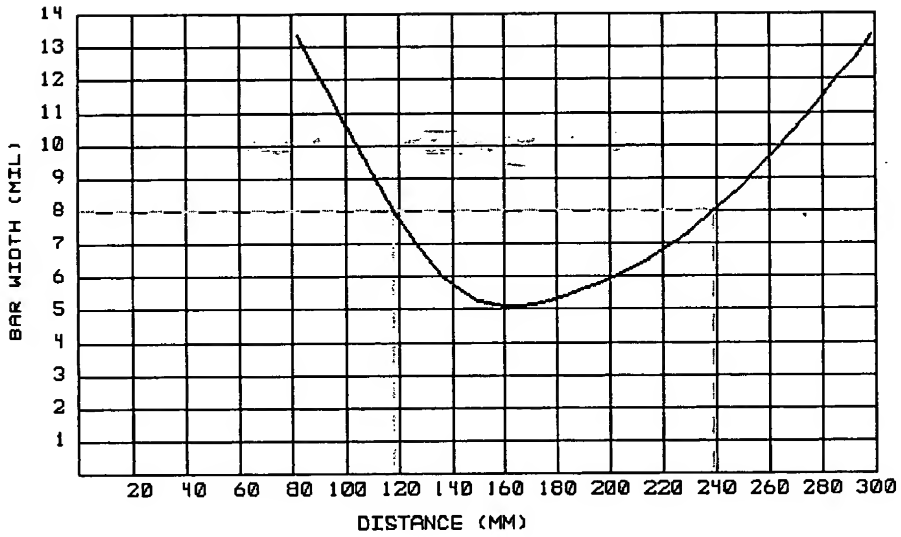


FIG. 4E

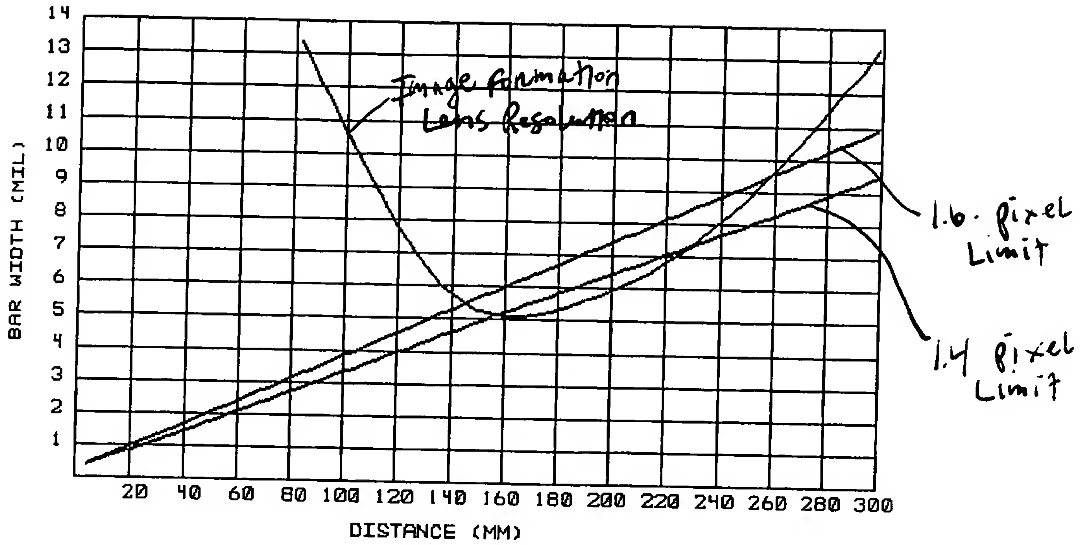


Image formation Lens Resolution
1.6. pixel Limit
1.4 pixel Limit
BAR WIDTH (MIL)
14
13
12
11
10
9
8
7
6
5
4
3
2
1
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
DISTANCE (MM)

FIG. 4F

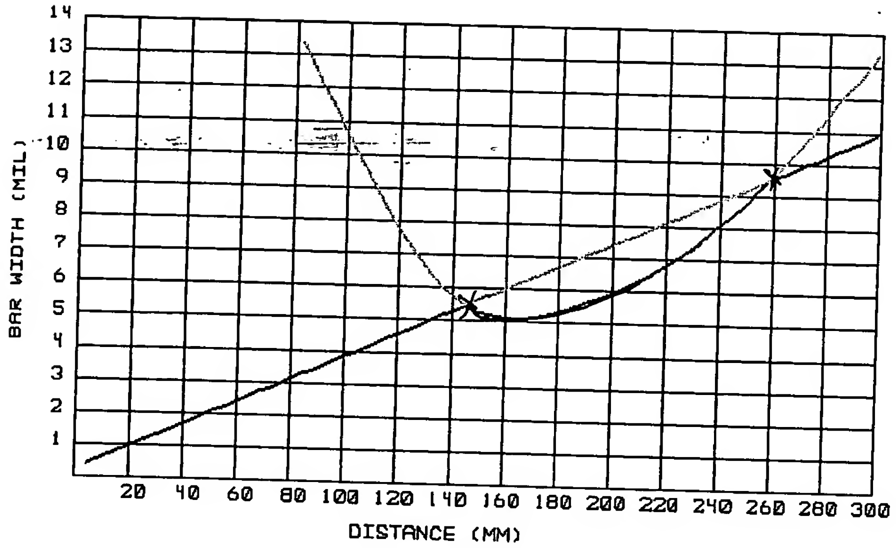

BAR WIDTH (MIL)
14
13
12
11
10
9
8
7
6
5
4
3
2
1
20 40 60 80 100 120 140 160 180 200 220 240 260 280 300
DISTANCE (MM)

FIG. 4G

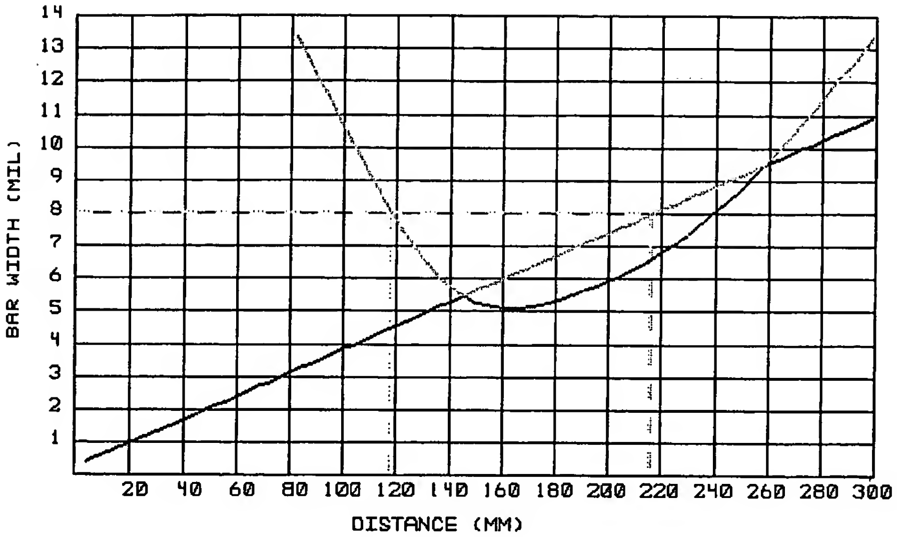

14
13
12
11
10
9
8
7
6
5
4
3
2
1
BAR WIDTH (MIL)
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
DISTANCE (MM)

FIG. 4H

DOF_PMAG.zpl

```
graphics
xmx=xmax()
xmn=xmin()
ymx=ymax()
ymn=ymin()
xwidth=xmx-xmn
ywidth=ymx-ymn
xleft=xmn+(0.1*xwidth)
xrigh=xmn+(0.95*xwidth)
ytopp=ymn+(0.05*ywidth)
ybott=ymn+(0.7*ywidth)

line xleft,ytopp,xrigh,ytopp
line xrigh,ytopp,xrigh,ybott
line xrigh,ybott,xleft,ybott
line xleft,ybott,xleft,ytopp

format 4.3
settextsize 140,80
gtext 0.68*xwidth,(0.85)*ywidth,0,"Wav :"
gtext 0.68*xwidth,(0.88)*ywidth,0,"WGT :"
for i=1,nwav(),1
        gtext (0.68+i*0.05)*xwidth,0.85*ywidth,0,$str(wavl(i))
        gtext (0.68+i*0.05)*xwidth,0.88*ywidth,0,$str(wwgt(i))
next
gtext 0.68*xwidth,(0.91)*ywidth,0,"Relative illumination : "
gtext 0.9*xwidth,(0.91)*ywidth,0,$str(reli(nfld()))
settextsize 90,50
input "Please input startpoint (mm):",start
if (start<=0) then input "Please input startpoint (mm):",start
input "Please input pixel size (um):",pix
if (pix<=0) then input "Please input pixel size (um):",pix
for i=start,start+150,10
        xpos=xleft+(i-start)/150*0.85*xwidth
        line xpos,ytopp,xpos,ybott
        format 3.0
        gtext xleft*0.85+(i-start)/150*0.85*xwidth,0.72*ywidth,0,$str(i)
next
settextsize 70,40
for i=1,14,1
        ypos=ytopp+i/14*.65*ywidth
        line xleft,ypos,xrigh,ypos
        format 3.0
        gtext 0.05*xwidth,ytopp*0.9+(i-1)/14*.65*ywidth,0,$str(14-i+1)
next


gtitle "The DOF and PMAG curve of current desigh"
gdate

format 12.6
oldthic=thic(0)

getsystemdata 2
settextsize 120,40
j=1
gtext xwidth*0.018,0.85*ywidth,0,"centering "
for i=1,nsur()-2,1
        if (gind(i)!=0.0)
                format 2.0
                gtext xwidth*0.10+(j-1)*0.07*xwidth,0.85*ywidth,0,$str(j)+":"
                gtext xwidth*0.12+(j-1)*0.07*xwidth,0.85*ywidth,0,":"
                format 4.2
```

FIG 4I1

DOF_PMAG.zpl

```
                if(curv(i)*curv(i+1)<0) then
centering=abso((sdia(i)*curv(i)+sdia(i+1)*curv(i+1)))
                if(curv(i)*curv(i+1)>0) then
centering=abso((sdia(i)*curv(i)-sdia(i+1)*curv(i+1)))
                gtext xwidth*0.13+(j-1)*0.07*xwidth,0.85*ywidth,0,$str(centering)
                j=j+1
        endif
next
format 4.2
settextsize 70,40
gtext xwidth*0.018,0.91*ywidth,0,"image space f/# : "+$str(vec2(8))
gtext xwidth*0.018,0.94*ywidth,0,"effective focal length : "+$str(vec2(7))
!color (3)
gtextcent ymn+(0.77*ywidth),"distance (mm)"
gtext xleft*0.32,0.5*ywidth,90,"bar width (mil)"

format 12.6
settextsize 100,40
minmtf=1
maxfreq=0
thic 0=start
update all
for k=0,200,0.2
        !i=nfld()
        for i=1,nfld(),1
                getmtf k,0,i,2,1,1
                !print vec1(0)
                !print vec1(1)
                if (vec1(0)<minmtf) then minmtf=vec1(0)
                if (vec1(1)<minmtf) then minmtf=vec1(1)
                if (minmtf<=0.3)
                        maxfreq=k
                        goto 1
                endif
        next
next
label 1
!color (1)

!output "1.txt" append

oldxpos=xleft+0/150*0.85*xwidth
oldypos=ytopp+(14-(1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000))/14*0.65*ywidth
switch=0
m=0
for j=start,start+150,3
        thic 0=j
        update all
        minmtf=1
        for k=m,200,0.3
                !i=nfld()
                for i=1,nfld(),1
                getmtf k,0,i,2,1,1
                if (vec1(0)<minmtf) then minmtf=vec1(0)
                if (vec1(1)<minmtf) then minmtf=vec1(1)
                if (minmtf<=0.3)
                        maxfreq=k
                        goto 2
                endif
                next
        next
        label 2
        if (maxfreq-5)>0
```

FIG. 4I2

DOF_PMAG.zpl

```
                m=maxfreq-10
        else
                m=0
        endif
        !print j,sdia(0),sdia(nsur()),maxfreq
        if ((switch==0) & (1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000<=13))
                !color (0)
                format 5.2
                a$="FOV for 10 mil: "+$str(2*sdia(0)) + " at "+$str(j-2)+" mm ; "
                gtext xwidth*0.018,0.97*ywidth,0,a$
                switch=1
                format 12.6
                !color(1)
        else
                if ((switch==1) &
(1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000>=13))
                        !color(0)
                        format 5.2
                        a$=$str(2*sdia(0))+" at "+$str(j-2)+" mm"
                        gtext xwidth*0.44,0.97*ywidth,0,a$
                        switch=0
                        format 12.6
                        goto 3
                        !color(1)
                endif
        endif
        newxpos=        xleft+(j-start)/150*0.85*xwidth

newypos=ytopp+(14-(1/(maxfreq/(sdia(0)/sdia(nsur())))*0.5/25.4*1000))/14*0.65*ywidth
        if ((14-14*(oldypos-ytopp)/0.65/ywidth)<14) then line
oldxpos,oldypos,newxpos,newypos
        oldxpos=newxpos
        oldypos=newypos
next
label 3
thic 0=start
update all
oldxpos=xleft+0/150*0.85*xwidth
oldxpos1=xleft+0/150*0.85*xwidth
oldypos=ytopp+(14-(0.5/((0.5/1.6/pix*1000)/(sdia(0)/sdia(nsur()))/25.4*1000))/14*0.
65*ywidth
oldypos1=ytopp+(14-(0.5/((0.5/1.4/pix*1000)/(sdia(0)/sdia(nsur()))/25.4*1000))/14*0
.65*ywidth
for j=start,start+150,4
        thic 0=j
        update all
        newxpos=xleft+(j-start)/150*0.85*xwidth
        newxpos1=xleft+(j-start)/150*0.85*xwidth

newypos=ytopp+(14-(0.5/((0.5/1.6/pix*1000)/(sdia(0)/sdia(nsur()))/25.4*1000))/14*0.
65*ywidth

newypos1=ytopp+(14-(0.5/((0.5/1.4/pix*1000)/(sdia(0)/sdia(nsur()))/25.4*1000))/14*0
.65*ywidth
        line oldxpos,oldypos,newxpos,newypos
        line oldxpos1,oldypos1,newxpos1,newypos1
        oldxpos=newxpos
        oldypos=newypos
        oldxpos1=newxpos1
        oldypos1=newypos1
next

thic 0=oldthic
```

FIG. 4I3

Multi-mode Illumination Subsystem

- Three modes of Illumination:

(1) Wide-Area for “near” object (0 mm – 100 mm)
(2) Wide-Area for “far” object (100 mm – 200 mm)
(3) Narrow-Area For “near” object (30 mm – 100 mm)

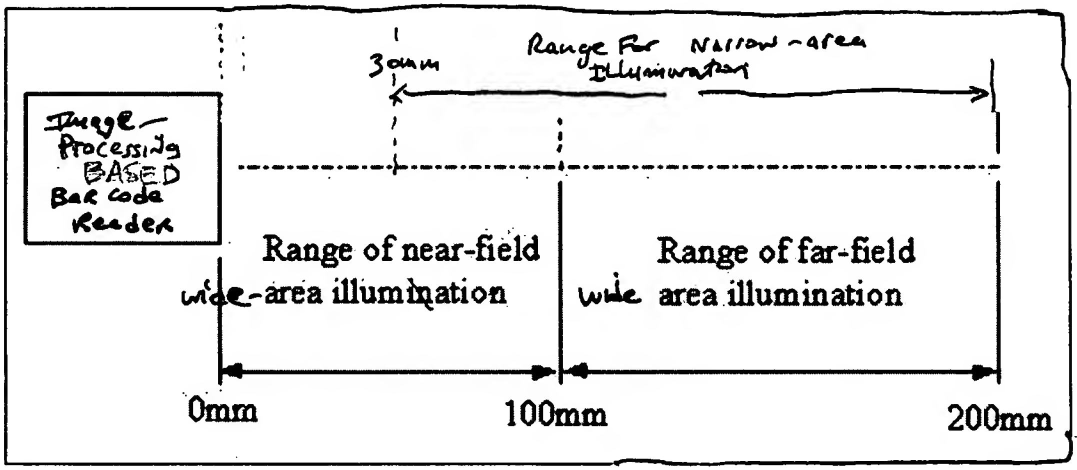


Image-Processing BASED Bar Code Reader
30mm
Range For Narrow-area Illumination
Range of near-field wide-area illumination
Range of far-field wide area illumination
0mm
100mm
200mm

FIG. 5A1

Illumination Design Goals FOR FIRST ILLUSTRATIVE EMBODIMENT

○ WIDE-AREA ILLUMINATION MODES

- Match FOV and DOF (45° , 200mm)
- Sufficient power density on target
  - Pixel value > 80 DN at far field center
- Achieve sufficient uniformity (center:edge = 2:1 max)
- Use as few LEDs as possible

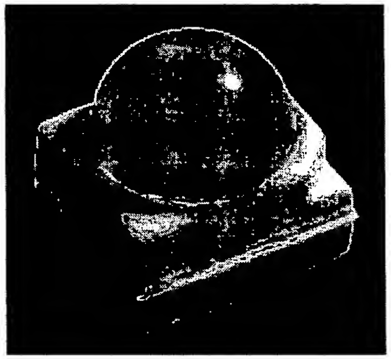
○ NARROW-AREA ILLUMINATION MODE

- Line usable beginning 40 mm from window
- Match FOV and DOF
- Suficient power density on target
- Sufficiently thin line
  - Height < 10 mm at far field

FIG. 5A2

LEDs For NARROW-AREA ILLUMINATION

- Linear illumination: Osram LS E655
  - 633 nm InGaAlP
  - 60° Lambertian emittance
  - 6.75 mW total output power (typical conditions)
  - $0.18 each in 50k



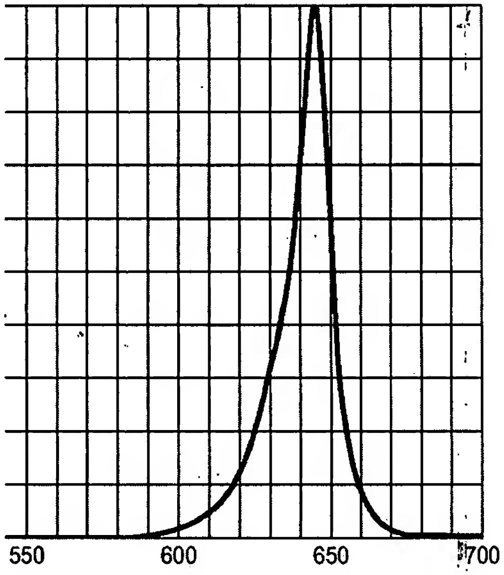


FIG. 5C1

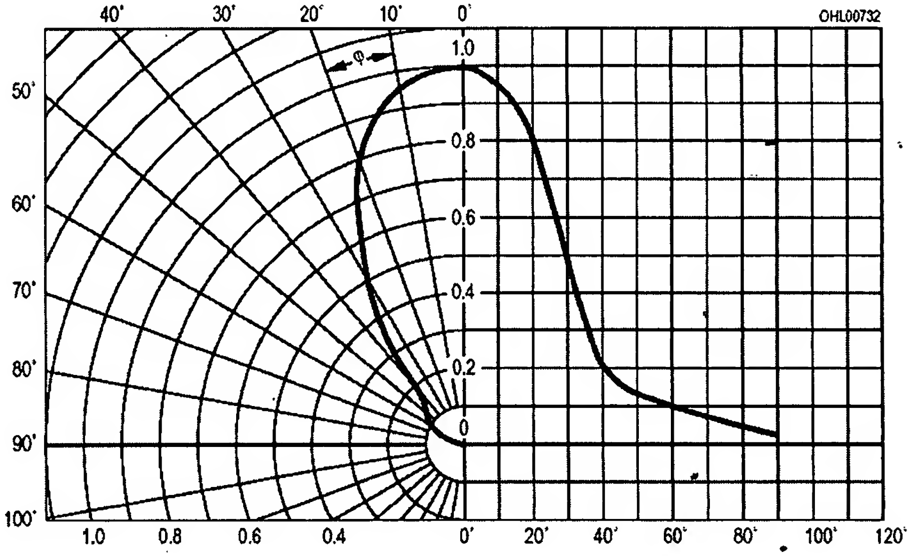


FIG. 5C2

LED Arrangements FOR NEAR-Field and FAR-Field WIDE AREA ILLUMINATION ARRAYS and NARROW-AREA ILLUMINATION ARRAY

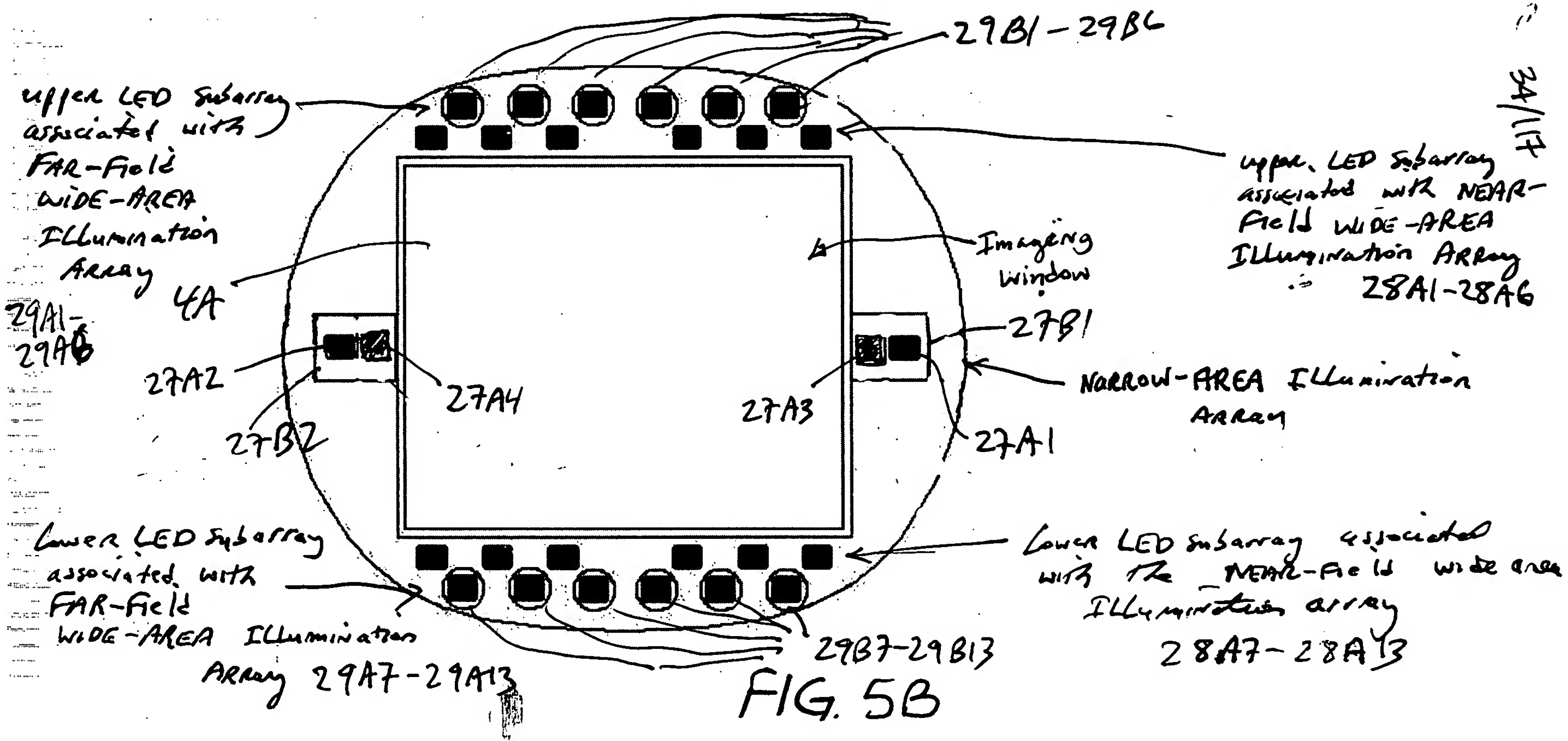

29B1 - 29B6
upper LED Subarray associated with FAR-Field WIDE-AREA ILLumination Array
29A1-29A6
upper LED Subarray associated with NEAR-Field WIDE-AREA ILLumination Array
28A1-28A6
4A
Imaging Window
27B1
27A2
27A4
27A3
NARROW-AREA ILLumination Array
27B2
27A1
Lower LED Subarray associated with FAR-Field WIDE-AREA ILLumination Array 29A7-29A13
Lower LED Subarray associated with the NEAR-Field wide area ILLumination array
29B7-29B13
28A7-28A13

FIG. 5B

Cylindrical Lenses FOR NARROW-AREA ILLUMINATION ARRAY

- First surface curved vertically to create line
- Second surface curved horizontally to control line height

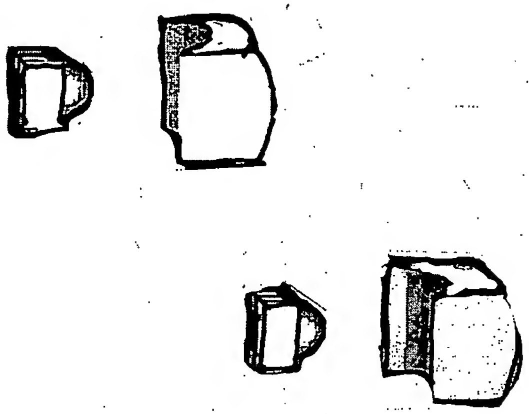


FIG. 5C3

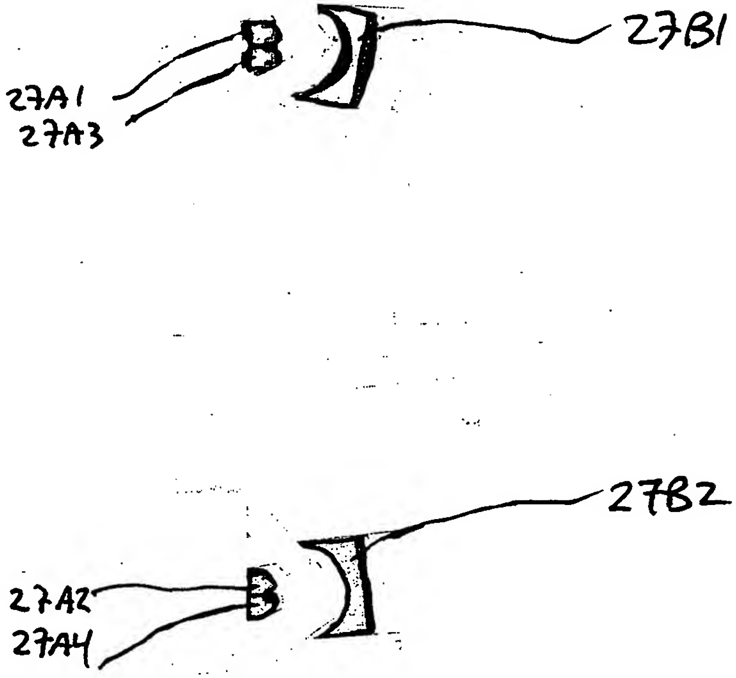


FIG. 5C4

Linear Illumination Profiles

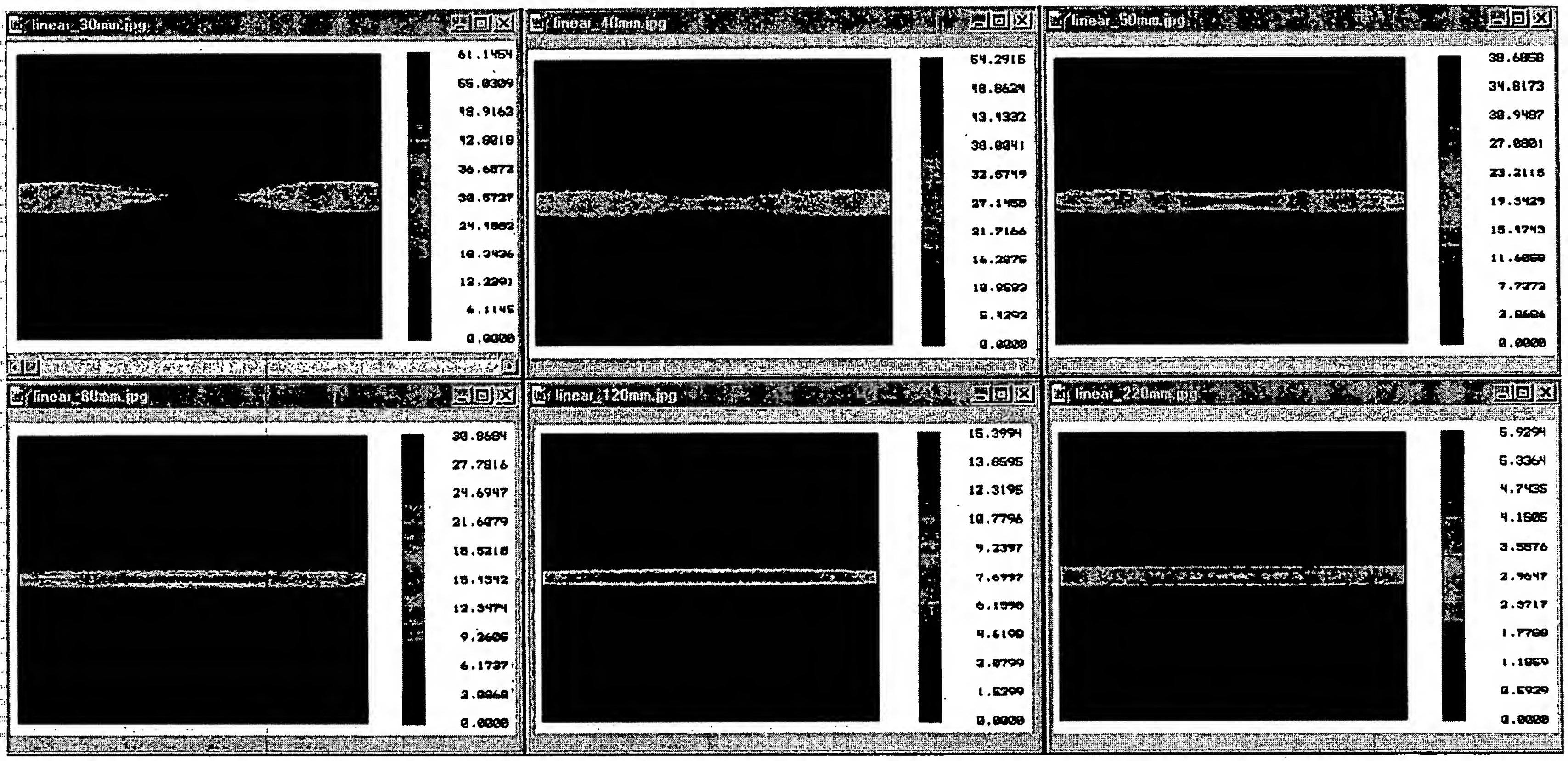
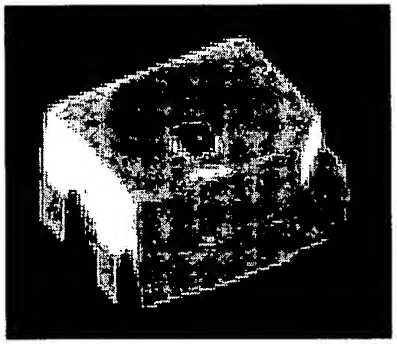


FIG. 5C5

Area LEDs

- Area illumination: Osram LS E67B
  - 633 nm InGaAlP
  - 120° Lambertian emittance
  - 11.7 mW total output power (typical conditions)
  - $0.18 each in 50k



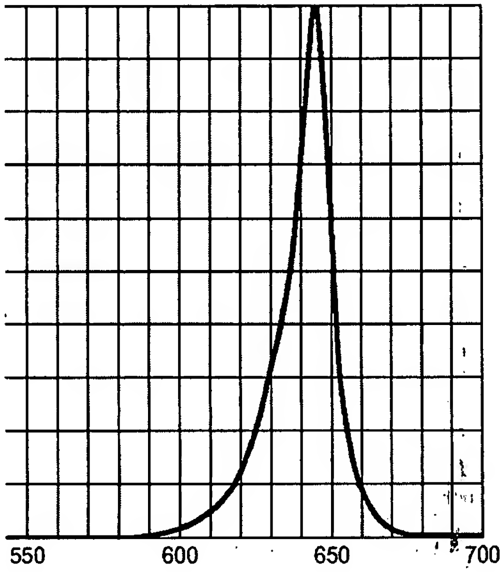


FIG. 5D1

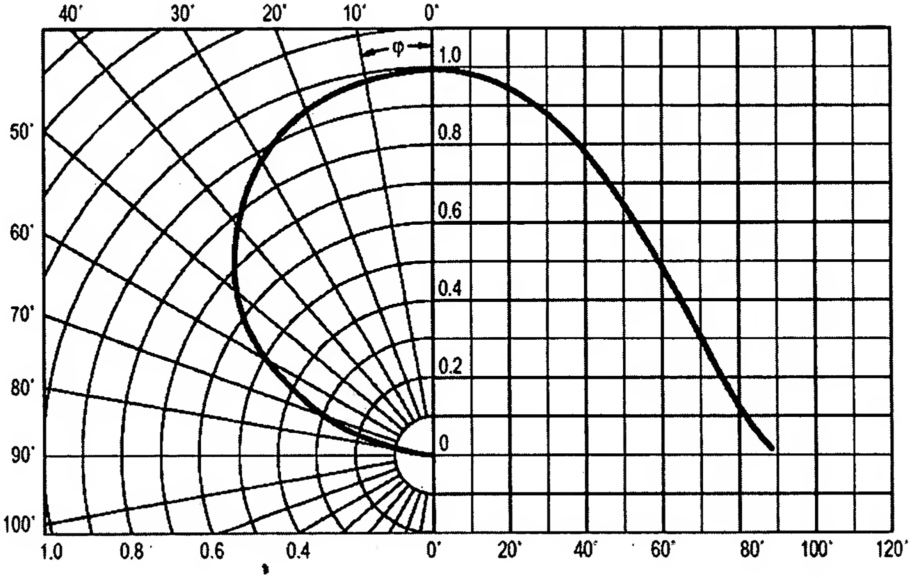


FIG. 5D2

Far Area Lenses

- Plano convex lenses in front of far field LEDs

- Light aimed by angling lenses
  - Even out distribution across FOV throughout DOF
  - Satisfy center:edge = 2:1 max criterion
  - Allows LEDs to be mounted flat

- All lenses CNCed in single piece of plastic

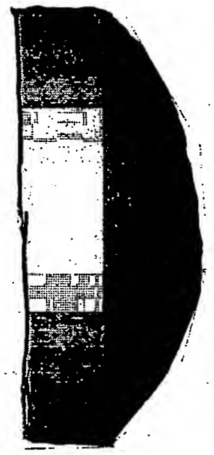


FIG.5D3

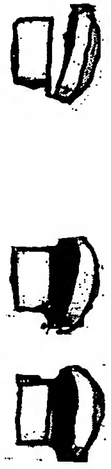


FIG5D4

WIDE-
Area Illumination Profiles (Near)

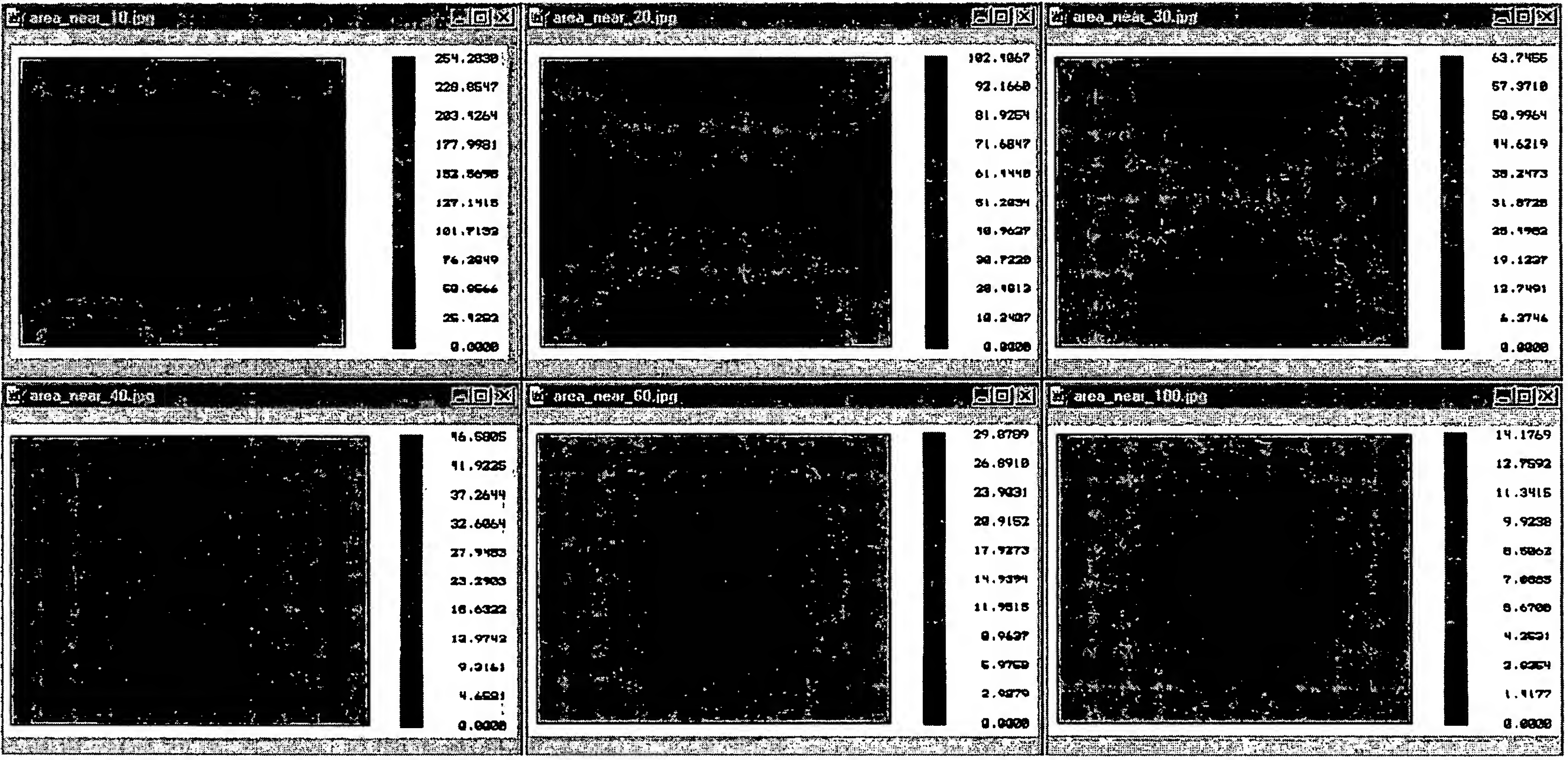
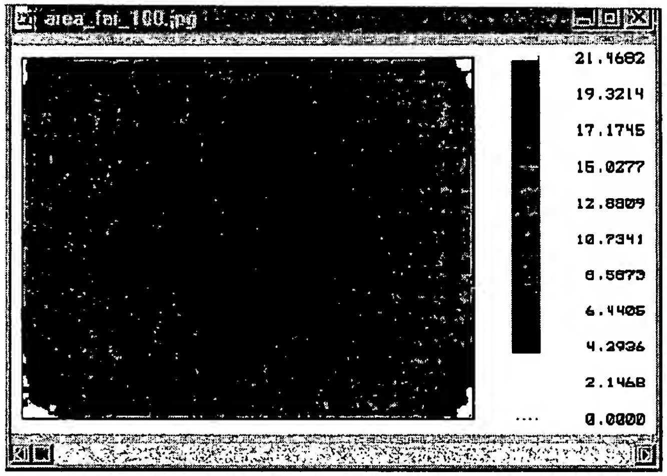
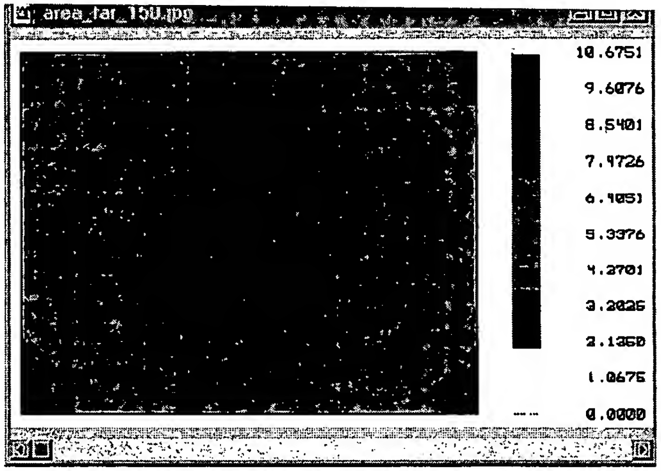


FIG. 5D5

WIDE-Area Illumination Profiles (Far)





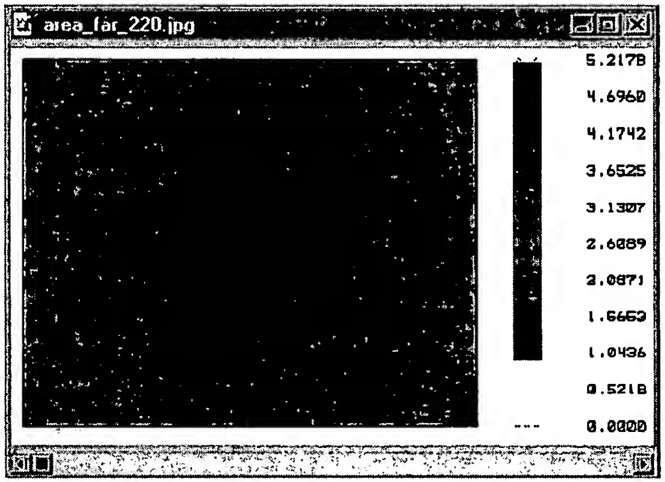


FIG. 5D6

Pixel Value Calculation

- Pixel value calculation for center of far field shows sufficient signal (> 80 DN)

description		value	unit
sensor power density	target power density	4	μW / mm^2
	surface reflectance	0.6	#
	optical transmittance	0.9	
	f-number	9	
	pixel power density	0.007	μW / mm^2
signal	CMOS internal gain	4.5	#
	amplification gain	20	dB
	integration time	5	ms
	sensor responsivity	1.8	V / (lx s)
	wavelength	633	nm
	photopic luminous efficiency	0.238	lm / W
	signal out of sensor	0.439	V
pixel value	A/D range max	1.3	V
	A/D range min	0.0	
	pixel value (0 - 255)	86	DN

FIG. 5D7

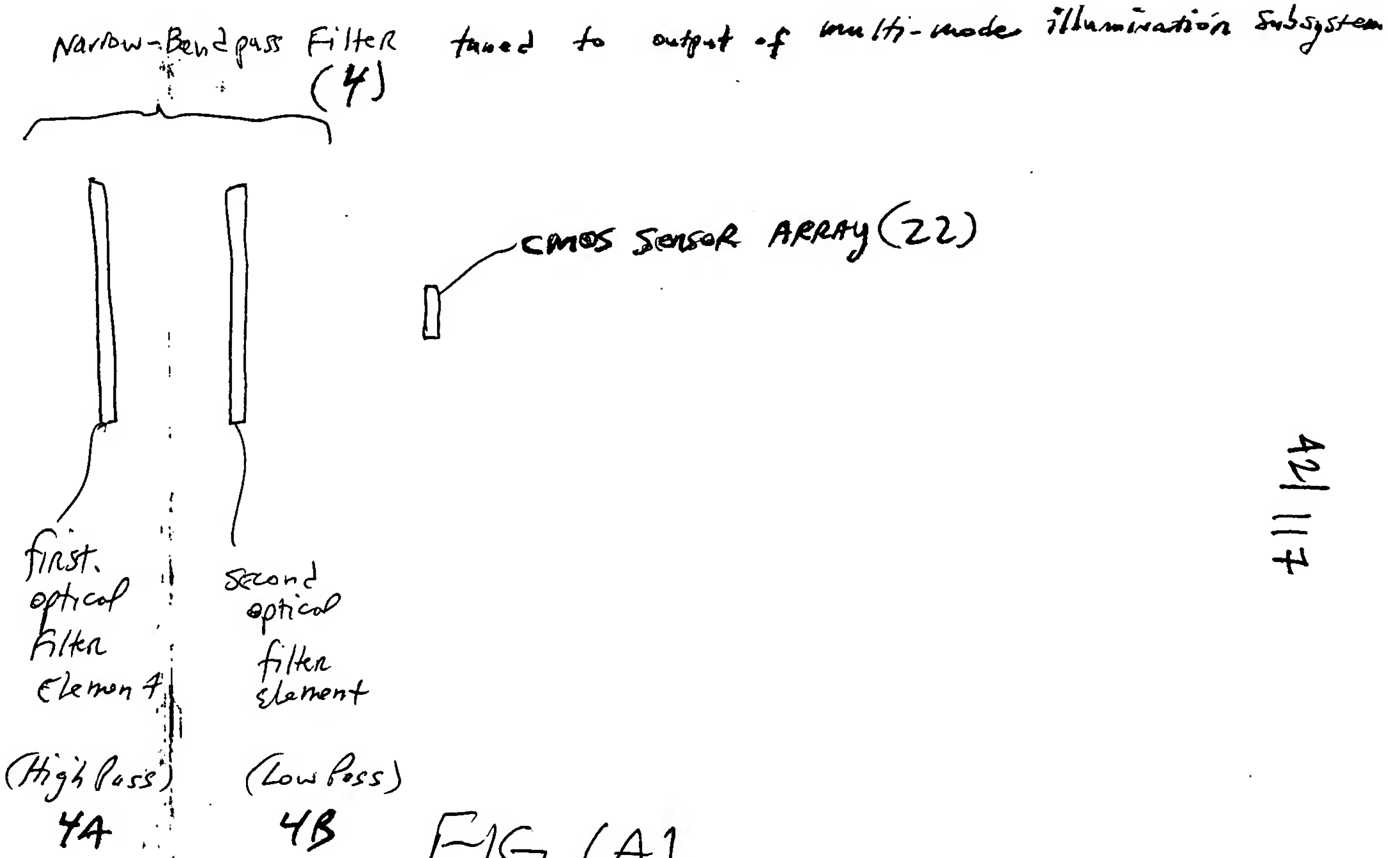

Narrow-Bendpass Filter (4) tuned to output of multi-mode illumination Subsystem
CMOS Sensor ARRAY (22)
first. optical filter Element 4
(High Pass)
4A
second optical filter element
(Low Pass)
4B

FIG. 6A1

Red Window and Low-Pass Filter Characteristics

- Must bandpass return light against ambient
  - Red window + low pass filter
  - Restricts range to 620nm – 700nm

low-pass

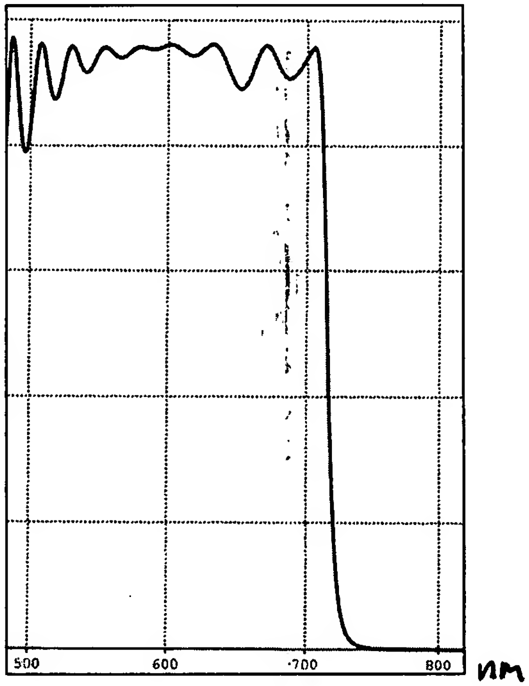


FIG. 6A2

(High-pass)
red window

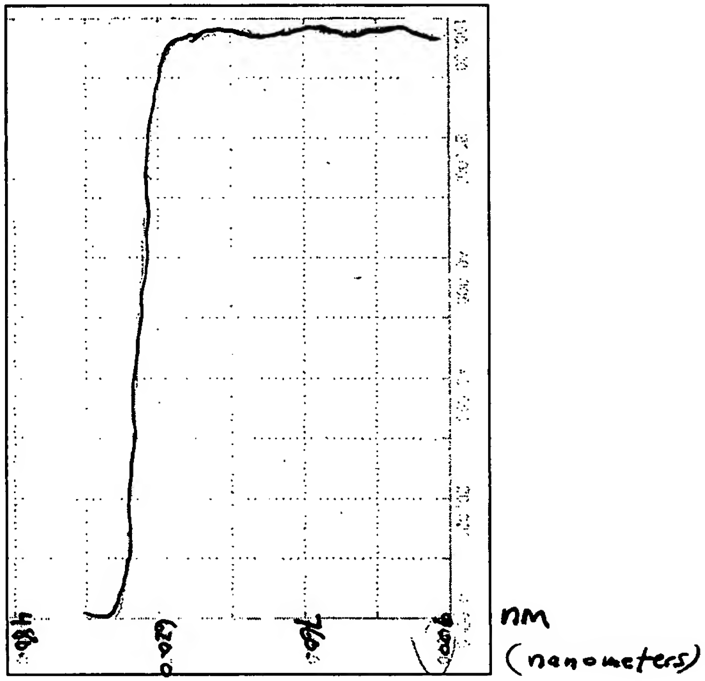
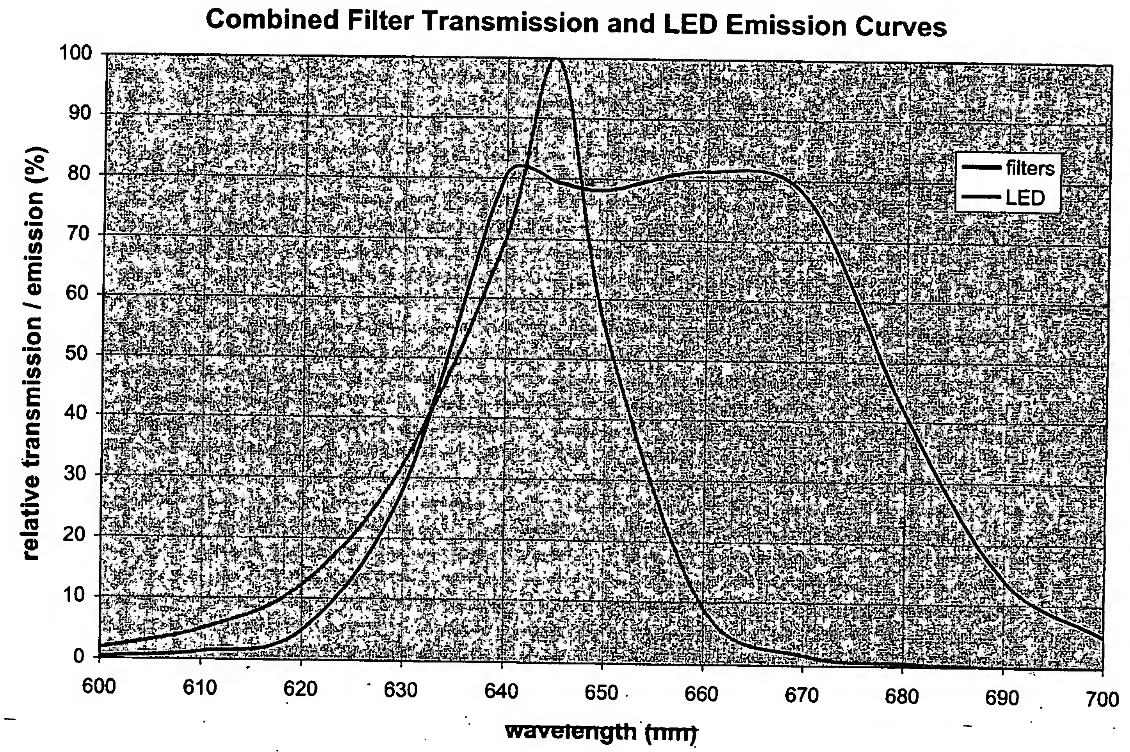


FIG. 6A3



(Δλ) Bandwidth of LED emission Signal ≈ 15 nmeters

FIG. 6A4

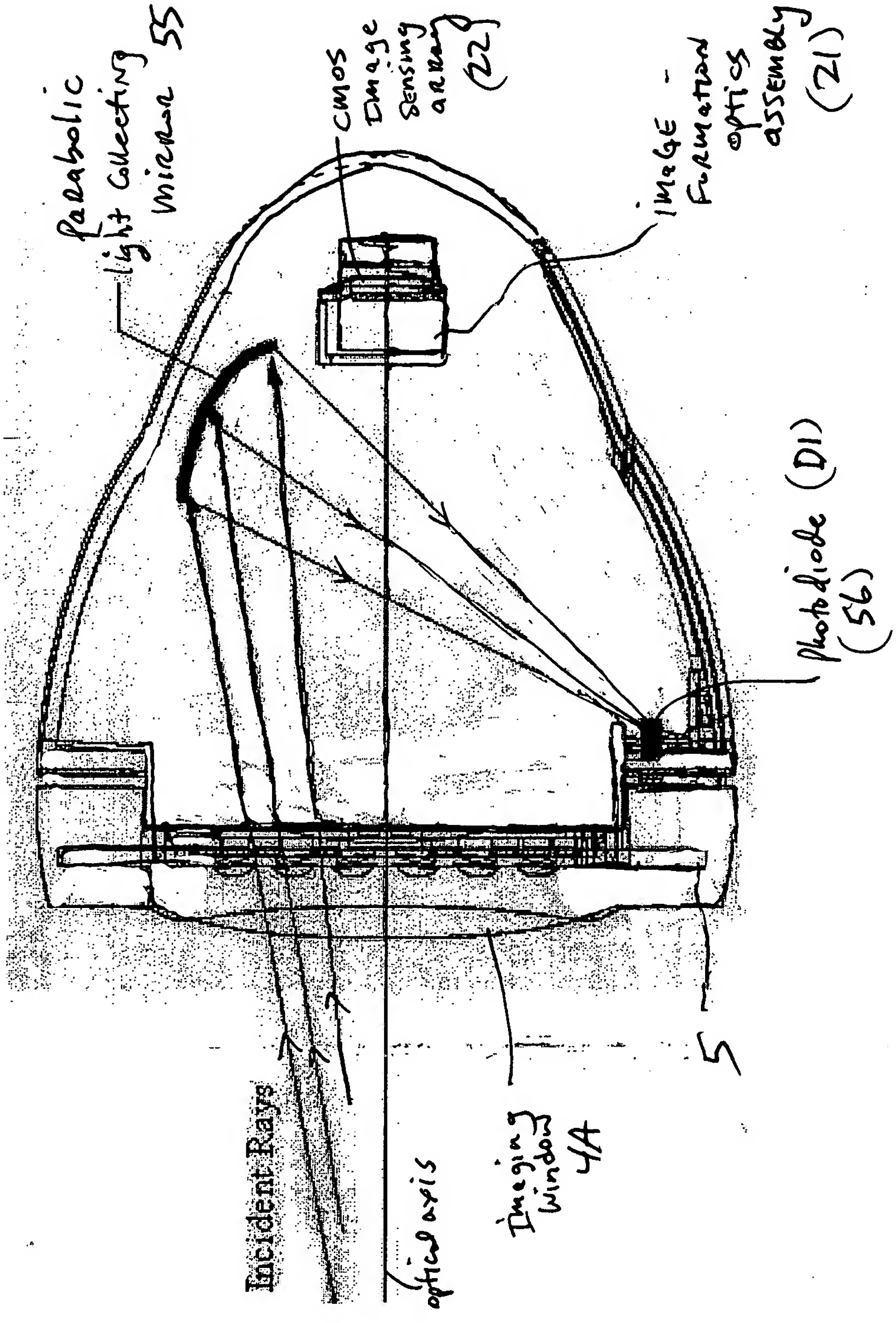

Parabolic light Collecting Mirror 55
CMOS Image sensing array (22)
Image Formation optics assembly (21)
Photodiode (D1) (56)
Incident Rays
optical axis
Imaging window 4A
5

FIG. 7A

Automatic LIGHT Exposure measurement and Illumination Control Subsystem (15)

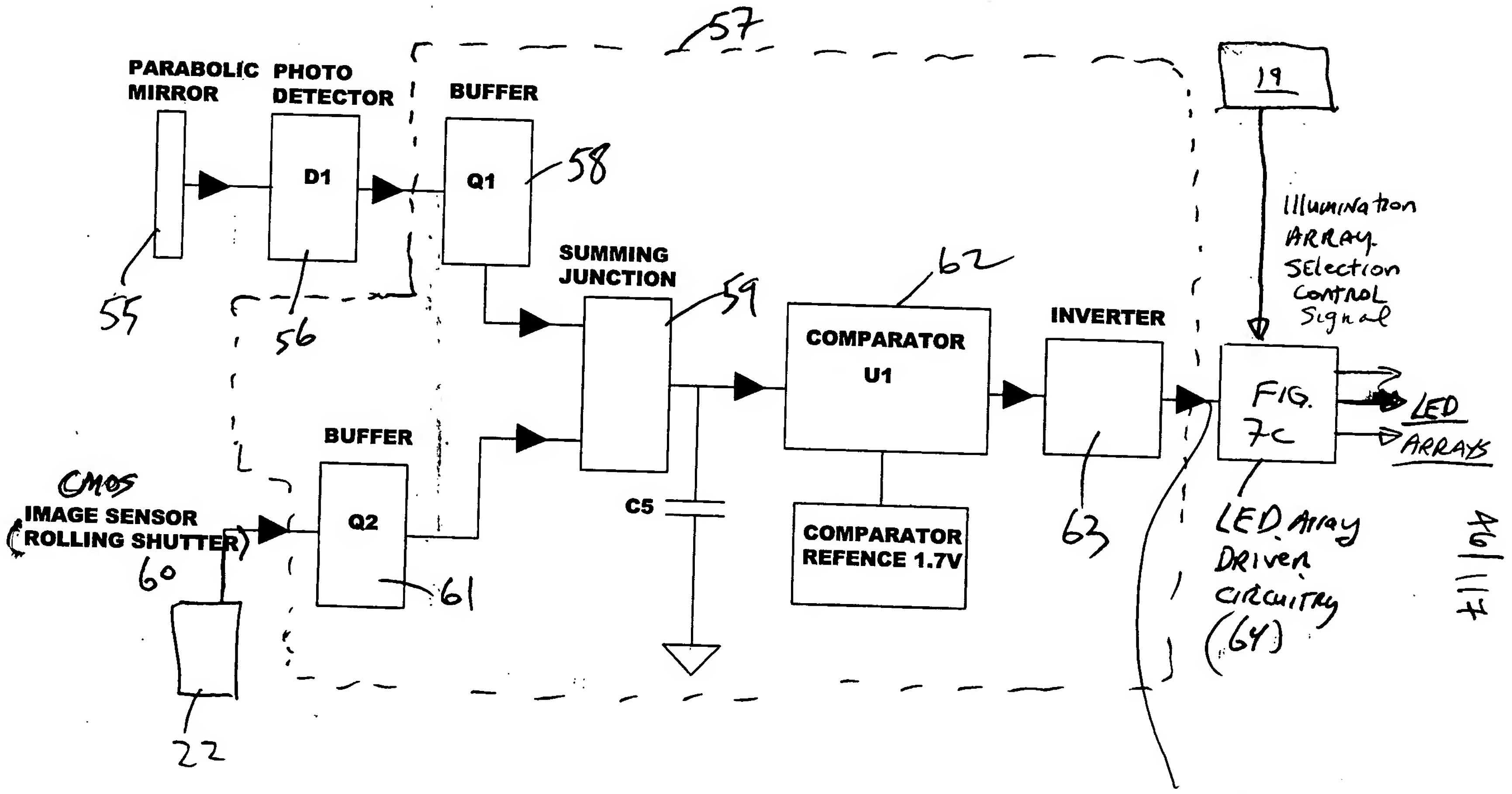


FIG. 7B

DRIVE CIRCUITRY FOR
LED ILLUMINATION ARRAYS

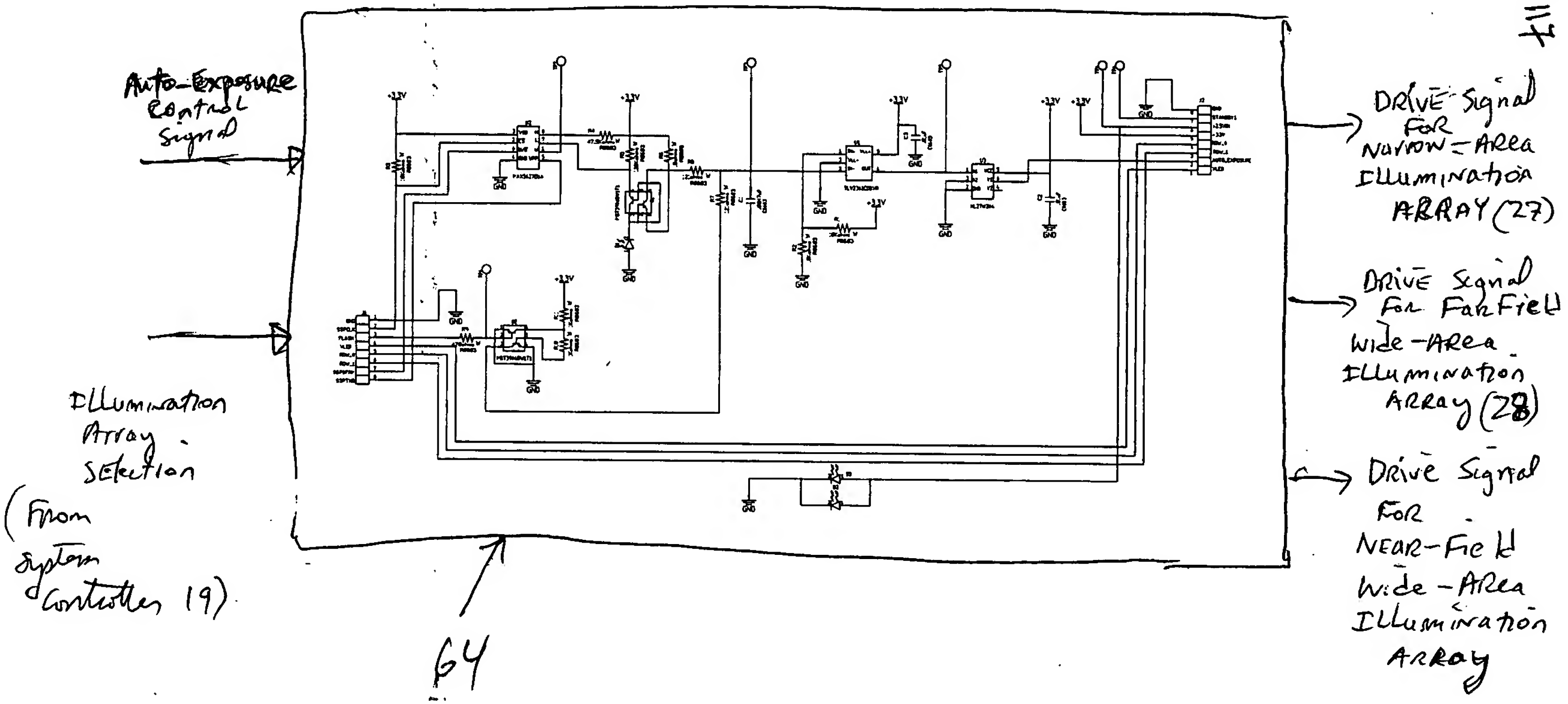
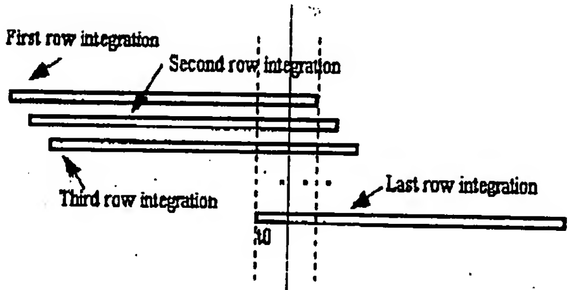

Auto-Exposure Control Signal
Illumination Array Selection
(From system controller 19)
DRIVE Signal FOR Narrow-Area Illumination ARRAY (27)
DRIVE Signal For FarField Wide-Area Illumination Array (28)
Drive Signal FOR NEAR-Field Wide-Area Illumination Array
64

FIG7C

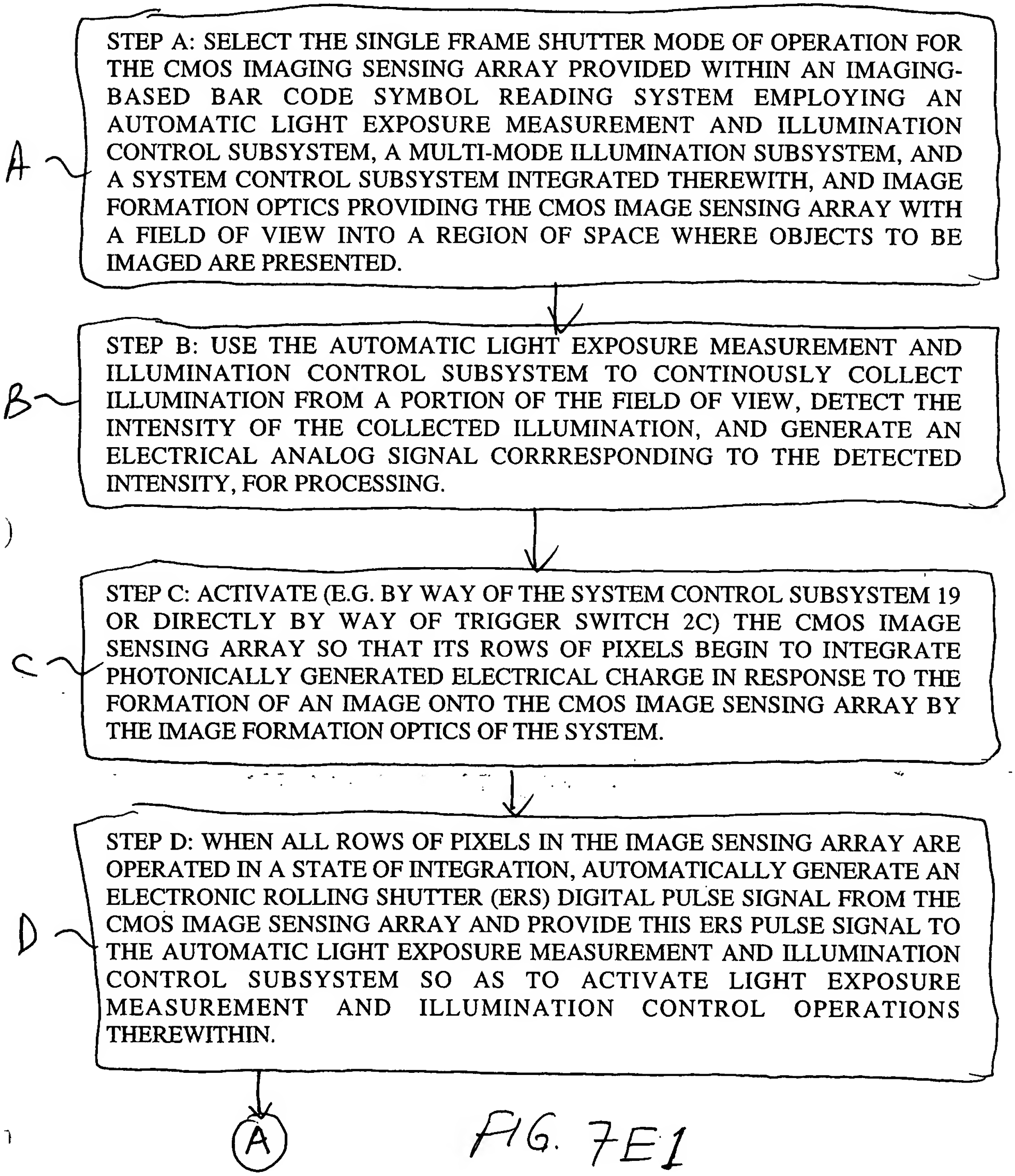
Global Exposure Control
Method of
Present Invention



b) Single Frame Shutter Mode

FIG. 7D

METHOD OF GLOBAL EXPOSURE CONTROL WITHIN A IMAGING-BASED BAR CODE SYMBOL READING SYSTEM



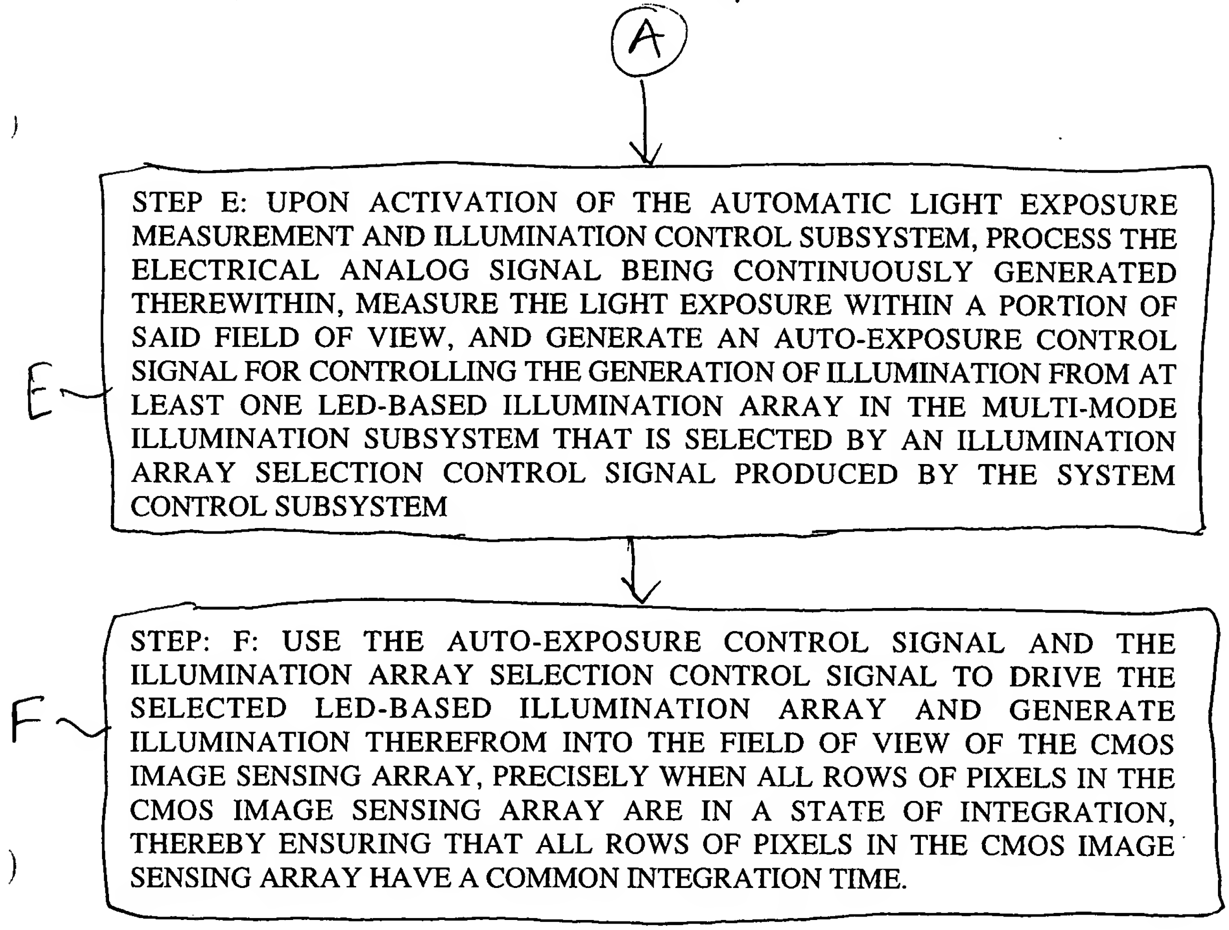
STEP A: SELECT THE SINGLE FRAME SHUTTER MODE OF OPERATION FOR THE CMOS IMAGING SENSING ARRAY PROVIDED WITHIN AN IMAGING-BASED BAR CODE SYMBOL READING SYSTEM EMPLOYING AN AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM, A MULTI-MODE ILLUMINATION SUBSYSTEM, AND A SYSTEM CONTROL SUBSYSTEM INTEGRATED THEREWITH, AND IMAGE FORMATION OPTICS PROVIDING THE CMOS IMAGE SENSING ARRAY WITH A FIELD OF VIEW INTO A REGION OF SPACE WHERE OBJECTS TO BE IMAGED ARE PRESENTED.

STEP B: USE THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM TO CONTINOUSLY COLLECT ILLUMINATION FROM A PORTION OF THE FIELD OF VIEW, DETECT THE INTENSITY OF THE COLLECTED ILLUMINATION, AND GENERATE AN ELECTRICAL ANALOG SIGNAL CORRRESPONDING TO THE DETECTED INTENSITY, FOR PROCESSING.

STEP C: ACTIVATE (E.G. BY WAY OF THE SYSTEM CONTROL SUBSYSTEM 19 OR DIRECTLY BY WAY OF TRIGGER SWITCH 2C) THE CMOS IMAGE SENSING ARRAY SO THAT ITS ROWS OF PIXELS BEGIN TO INTEGRATE PHOTONICALLY GENERATED ELECTRICAL CHARGE IN RESPONSE TO THE FORMATION OF AN IMAGE ONTO THE CMOS IMAGE SENSING ARRAY BY THE IMAGE FORMATION OPTICS OF THE SYSTEM.

STEP D: WHEN ALL ROWS OF PIXELS IN THE IMAGE SENSING ARRAY ARE OPERATED IN A STATE OF INTEGRATION, AUTOMATICALLY GENERATE AN ELECTRONIC ROLLING SHUTTER (ERS) DIGITAL PULSE SIGNAL FROM THE CMOS IMAGE SENSING ARRAY AND PROVIDE THIS ERS PULSE SIGNAL TO THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM SO AS TO ACTIVATE LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL OPERATIONS THEREWITHIN.

FIG. 7E1



STEP E: UPON ACTIVATION OF THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND ILLUMINATION CONTROL SUBSYSTEM, PROCESS THE ELECTRICAL ANALOG SIGNAL BEING CONTINUOUSLY GENERATED THEREWITHIN, MEASURE THE LIGHT EXPOSURE WITHIN A PORTION OF SAID FIELD OF VIEW, AND GENERATE AN AUTO-EXPOSURE CONTROL SIGNAL FOR CONTROLLING THE GENERATION OF ILLUMINATION FROM AT LEAST ONE LED-BASED ILLUMINATION ARRAY IN THE MULTI-MODE ILLUMINATION SUBSYSTEM THAT IS SELECTED BY AN ILLUMINATION ARRAY SELECTION CONTROL SIGNAL PRODUCED BY THE SYSTEM CONTROL SUBSYSTEM

STEP: F: USE THE AUTO-EXPOSURE CONTROL SIGNAL AND THE ILLUMINATION ARRAY SELECTION CONTROL SIGNAL TO DRIVE THE SELECTED LED-BASED ILLUMINATION ARRAY AND GENERATE ILLUMINATION THEREFROM INTO THE FIELD OF VIEW OF THE CMOS IMAGE SENSING ARRAY, PRECISELY WHEN ALL ROWS OF PIXELS IN THE CMOS IMAGE SENSING ARRAY ARE IN A STATE OF INTEGRATION, THEREBY ENSURING THAT ALL ROWS OF PIXELS IN THE CMOS IMAGE SENSING ARRAY HAVE A COMMON INTEGRATION TIME.

FIG. 7E2

IR object Presence & Range detection

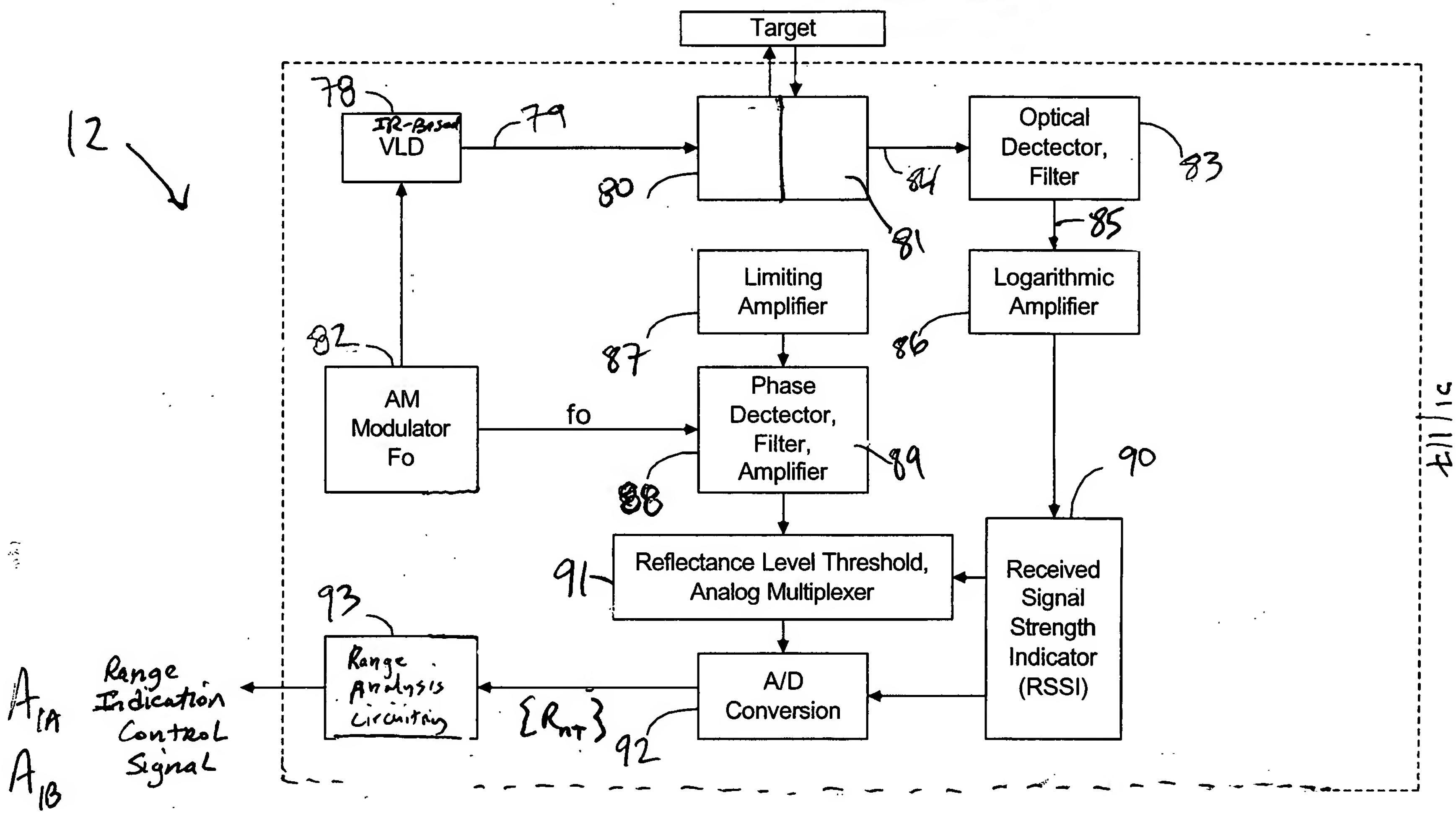

12
Target
78
IR-Based VLD
79
80
81
84
Optical Dectector, Filter
83
85
Logarithmic Amplifier
86
Limiting Amplifier
87
82
AM Modulator Fo
fo
Phase Dectector, Filter, Amplifier
88
89
90
Received Signal Strength Indicator (RSSI)
91
Reflectance Level Threshold, Analog Multiplexer
A/D Conversion
92
{R_NT}
93
Range Analysis Circuitry
Range Indication Control Signal
A1A
A1B
(To System Control Subsystem (19))

FIG. 8

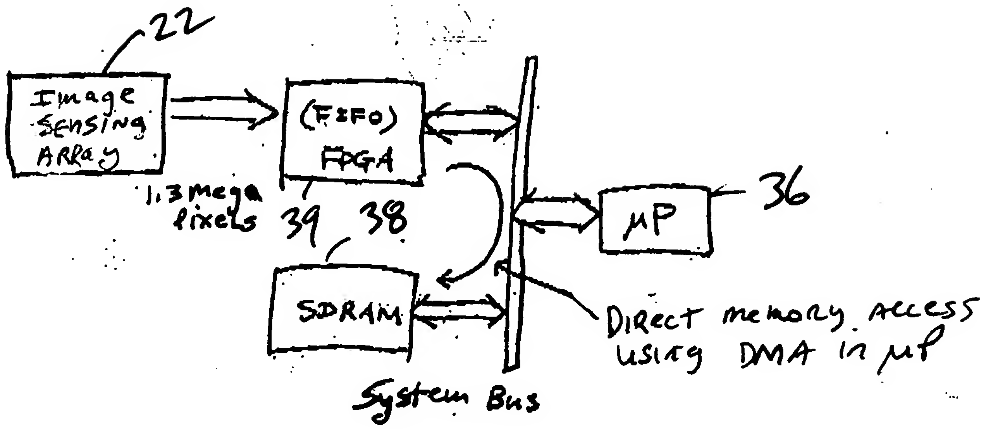

22
Image Sensing Array
(FIFO) FPGA
1.3 mega pixels
39
38
SDRAM
µP
36
Direct memory access using DMA in µP
System Bus

FIG. 9.

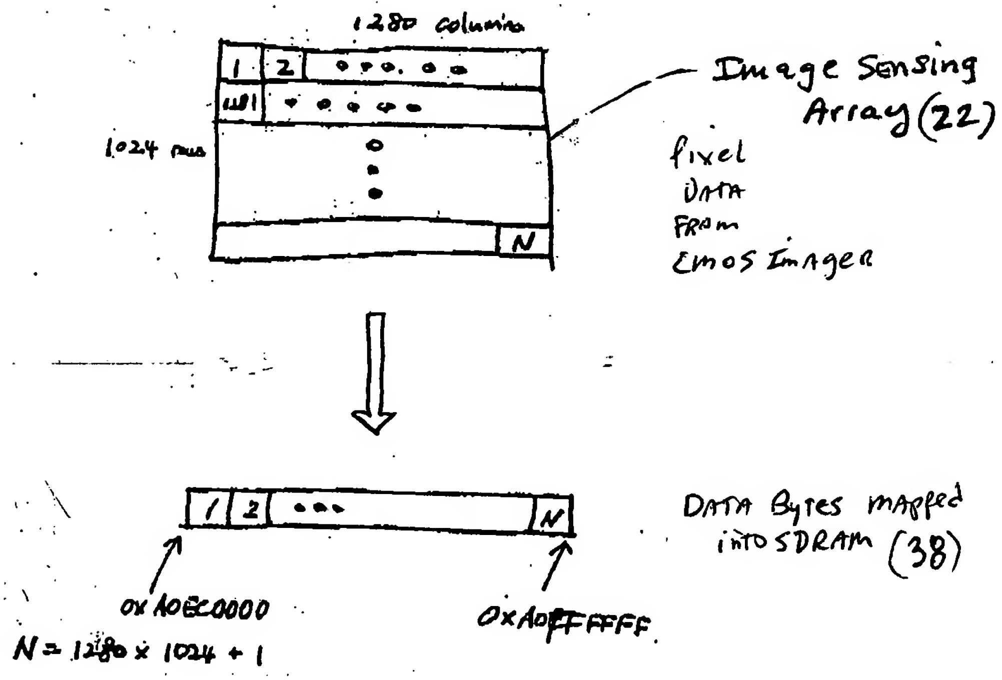
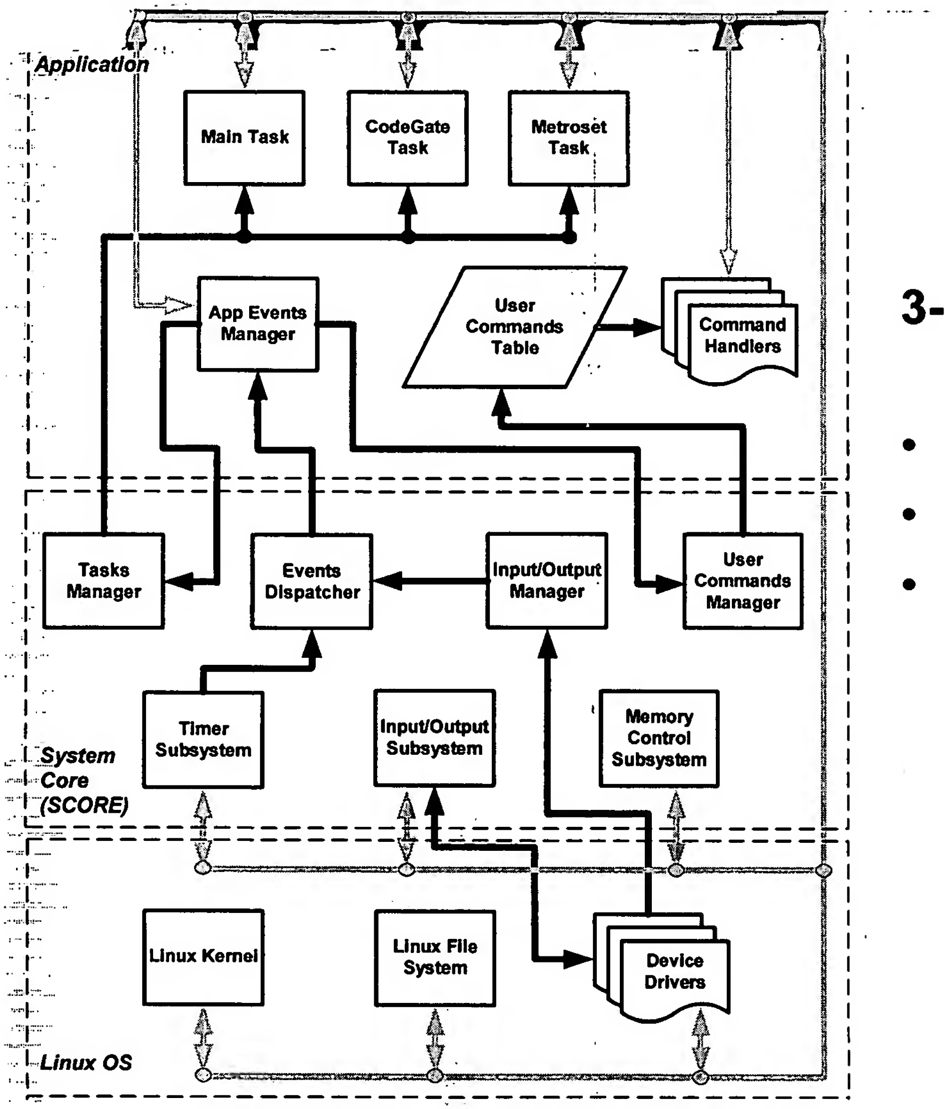

1280 columns
1
2
1281
1024 rows
N
Image Sensing Array (22)
Pixel Data From CMOS Imager
1
2
N
DATA Bytes mapped into SDRAM (38)
0x A0EC0000
0xA0FFFFFF
N = 1280 x 1024 + 1

FIG. 10

Software Block Diagram



3-Tier Software Architecture:

- **Linux OS**
- **System Core (SCORE) Software**
- **Product Application Software**

FIG. 11

Events Dispatcher

Provides a means of signaling and delivering events to the App Events Manager

(pointer to App Events Manager is provided at the SCORE initialization)

```
int
ScoreSignalEvent(int event_id,    /* Input:  event id */
          void * p_par);          /* Input:  pointer to the event's parameters */
```

App Events Manager is responsible for processing the event: It can start a new task, or stop currently running task, or do something or nothing and simply ignore the event.

FIG. 12A

Examples of System-Defined Events

SCORE_EVENT_POWER_UP

Signals the completion of the system start-up. No parameters.

SCORE_EVENT_TIMEOUT

Signals the timeout of the logical timer. Parameter: pointer to timer id.

SCORE_EVENT_UNEXPECTED_INPUT

Signals that the unexpected input data is available. Parameter: pointer to connection id.

SCORE_EVENT_TRIG_ON

Signals that the user pulled the trigger. No parameters.

SCORE_EVENT_TRIG_OFF

Signals that the user released the trigger. No parameters.

SCORE_EVENT_OBJECT_DETECT_ON

Signals that the object is positioned under the camera. No parameters.

SCORE_EVENT_OBJECT_DETECT_OFF

Signals that the object is removed from the field-of view of the camera. No parameters.

SCORE_EVENT_EXIT_TASK and **SCORE_EVENT_ABORT_TASK**

Signal the end of the task execution. Parameter: pointer to the UTID.

FIG.12B

Tasks Manager

Provides a means of executing and stopping application specific tasks (threads)

```
typedef void *                                /* Return: pointer to the set of returned parameters */
(*TASK_FUNC)(void *params);                   /* Input:  set of input parameters */

int                                           /* Return: 0 if successful, otherwise error code */
ScoreStartTask(TASK_FUNC task_func,           /* Input:  pointer to the task's main function */
      int task_id,                            /* Input:  id assigned to the task by application */
     void *task_params,                       /* Input:  parameters passed to the task's main function */
     int task_owner,                          /* Input:  connection id of the task's owner */
     int task_priority,                       /* Input:  task's priority (must be 0 for now) */
     size_t stacksize,                        /* Input:  size of the stack, or 0 for default size */
     size_t heapsize,                         /* Input:  size of the heap, or 0 for default size */
     UTID *p_utid);                           /* Output: unique task identifier */

BOOL      /* Return: TRUE if it kills the task, or FALSE if the task was not found */
ScoreKillTask(UTID pthread_id)                /* Input:  unique task identifer */
```

FIG 12C

Input / Output Manager

- High priority thread running in the background and monitoring activities of the external devices and user connections
- Signals appropriate events to the application when such activities are detected

FIG. 12D

Input / Output Subsystem

Provides a means of creating and deleting input/output connections...

```
int                                         /* Return: connection id if successful, otherwise (-1) */
ScoreIomngrCreateConnection(int conn_type,          /* Input:  connection type */
    int fd,                                 /* Input:  file descriptor of a device or a socket */
    int conn_state,                         /* Input:  initial state of the connection, the value controlled by application */
    void *properties);                      /* Input:  pointer to the connection properties */

int                                         /* Return: connection id if successful, otherwise (-1) */
ScoreInitRS232(char *full_name,                     /* Input:  full name of the device, such as "/dev/ttyS0" */
        RS232_PROP *rs232_prop);                    /* Input: RS232 parameters */
```

FIG. 12E1

Input / Output Subsystem

...and communicating with the outside world

```
int                                 /* Return: number of bytes received */
ScoreIomngrGetData(int connection_id,       /* Input:  connection id, or -1 for the task owner */
          char *input_buffer,       /* Input:  pointer to the input buffer */
          int min_len,              /* Input:  minimum number of bytes to receive */
          int max_len,              /* Input:  maximum number of bytes to receive */
          BOOL echo,                /* Input:  TRUE if data should be echoed back to device, otherwise FALSE */
          int timeout_ms);          /* Input:  If not 0, number of milliseconds to wait */

int                                 /* Return: 0 if successful, or (-1) in case of error */
ScoreIomngrSendData(int connection_id,      /* Input:  connection id */
          char *p_data,             /* Input:  pointer to the data buffer */
          int len);                 /* Input:  number of bytes to send */

void
ScoreIomngrSendStream(int stream_type,      /* Input:  type of output stream */
           char *p_data,                    /* Input:  pointer to the data buffer */
           int len);                        /* Input:  number of bytes to send */
```

FIG. 12E2

Timer Subsystem

Provides a means of creating, deleting...

```
int                                  /* Return: timer id if successful, otherwise (-1) */
ScoreCreateTimer(int flags);         /* Input:  optional SCORE_TIMER_CONTINUOUS */

void
ScoreDeleteTimer(int timer_id);      /* Input:  timer id, must be >= 0 */

int                                  /* Return: 0 if successful, otherwise (-1) */
ScoreStartTimer(int timer_id,        /* Input:  timer id */
        int time_ms);                /* Input:  timer value, in ms */

int                                  /* Return: 0 if successful, otherwise (-1) */
ScoreStopTimer(int timer_id);        /* Input:  timer id */
```

FIG.12 F1

Timer Subsystem

...and utilizing logical timers

```
BOOL                              /* Return: TRUE if the timer timed out, otherwise FALSE */
ScoreTimerTimedOut(int timer_id);           /* Input:  timer id */

int                        /* Return: time (in ms) left before the timer times out, or (-1) in case of error */
ScoreGetTimeLeft(int timer_id);             /* Input:  timer id */

int             /* Return: time (in ms) gone since the timer has been started (or restarted), or (-1) in case of error */
ScoreGetTime(int timer_id);                 /* Input:  timer id */

BOOL                       /* Return: TRUE if timer is stopped, otherwise FALSE */
ScoreIsTimerStopped(int timer_id);          /* Input:  timer id */
```

FIG. 12F2

Memory Control Subsystem

Provides a thread-level dynamic memory management (the interfaces fully compatible with standard dynamic memory management functions)...

```
void *                        /* Return: pointer to the allocated memory if successful, otherwise NULL */
ScoreMalloc(size_t size);     /* Input:  size, in bytes, of the needed memory */

void
ScoreFree(void *mem);         /* Input:  pointer to the memory to be freed */
```

FIG.12G1

Memory Control Subsystem

...as well as a means of buffering the data

```
int                                                    /* Return: 0 if successful */
ScoreCreateOutpMem(SCORE_OUTP_MEM *p_outp_mem);        /* Input: pointer to buffered memory structure */

void
ScoreDestroyOutpMem(SCORE_OUTP_MEM  *p_outp_mem);      /* Input: pointer to buffered memory structure */

int                                           /* Return: 0 if successful */
ScoreWriteToOutpMem (SCORE_OUTP_MEM *p_outp_mem,       /* Input: pointer to buffered memory structure */
      void *p_data,                           /* Input: pointer to the data to be buffered up for output */
      size_t len);                            /* Input: size of the data, in bytes */

int                                           /* Return: 0 if successful */
ScoreSendDataFromOutpMem(int connection_id,   /* Input: id of the connection to send the data to */
      SCORE_OUTP_MEM  *p_outp_mem);            /* Input: pointer to buffered memory structure */

int                                           /* Return: 0 if successful */
ScoreSendStreamFromOutpMem(int stream_type,   /* Input:  type of output stream */
      SCORE_OUTP_MEM *p_outp_mem);            /* Input: pointer to buffered memory structure */
```

FIG.12G2

User Commands Manager

Provides a standard way of entering user commands and executing application modules responsible for handling them

(pointer to User Commands Table is provided at the SCORE initialization)

```
int
ScoreCmdManager(void *params);

rc = ScoreStartTask(ScoreCmdManager,      /* Input:  user command manager task */
       CMDMNGR_TASK_ID,                   /* Input:  id assigned to the commands manager */
       NULL,
       0,
       connection_id,                     /* Input:  connection id of the owner */
       0,                                 /* Input:  priority */
       (64 * 1024),                       /* Input:  stack size */
       (512 * 1024),                      /* Input:  heap size */
       &cmdmngr_utid);                    /* Output: unique task identifier */
```

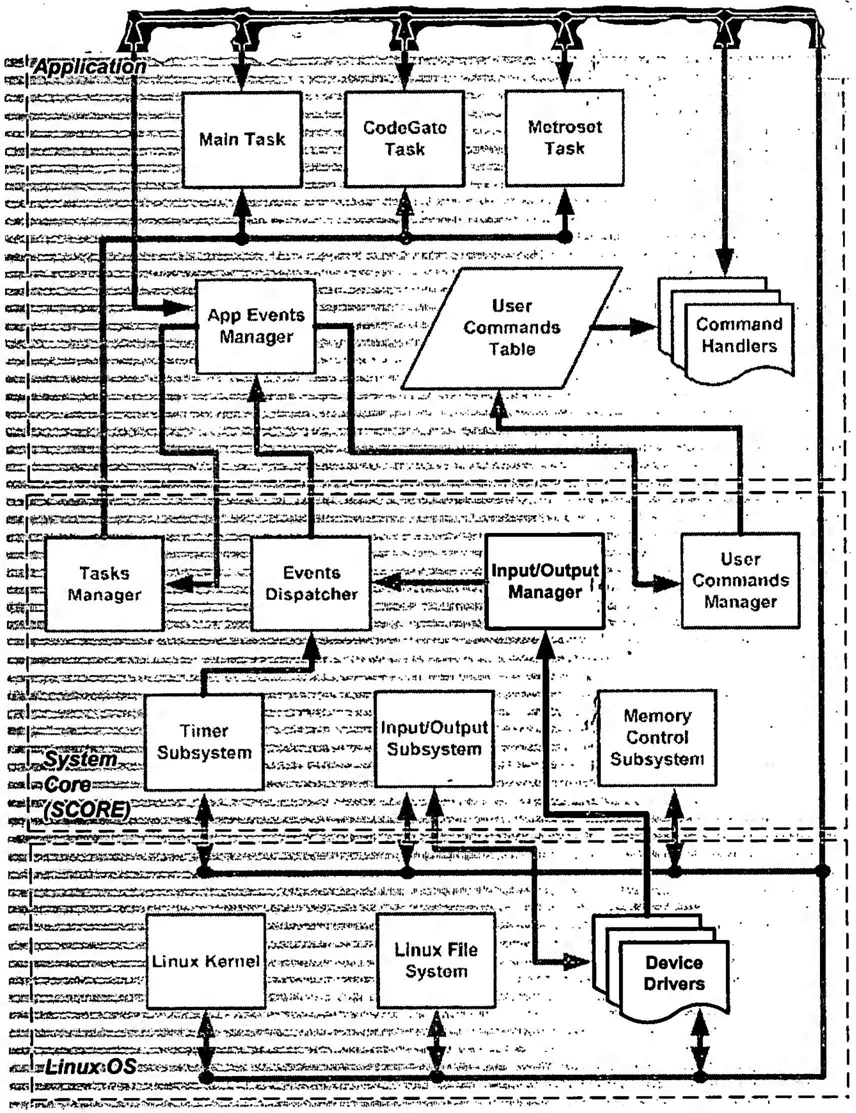
FIG. 12H

Device Drivers

- Trigger driver -- establishes software connection with the hardware trigger
- Image acquisition driver -- implements image acquisition functionality
- IR driver -- implements object detection functionality

FIG. 12I

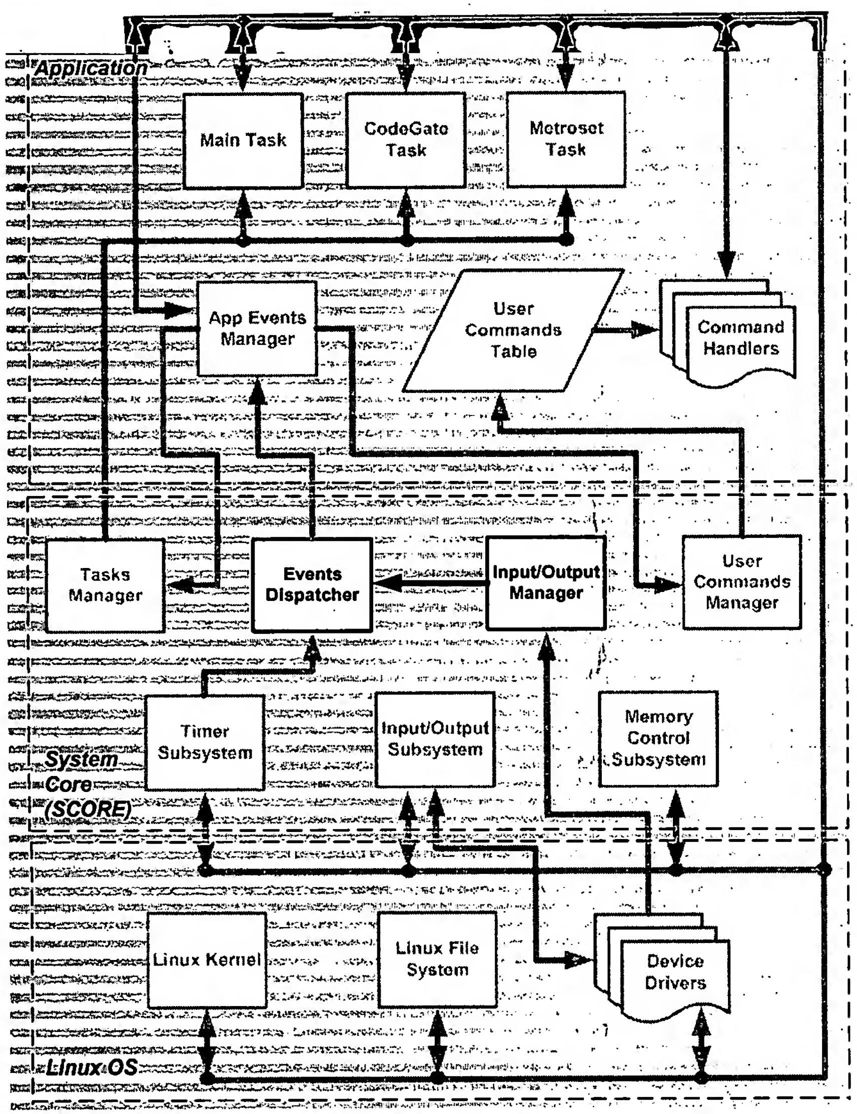
Example of Flow of Events



- User points the scanner towards a barcode label
- Object is detected
- The IR device driver wakes up the Input/Output Manager

FIG. 13A

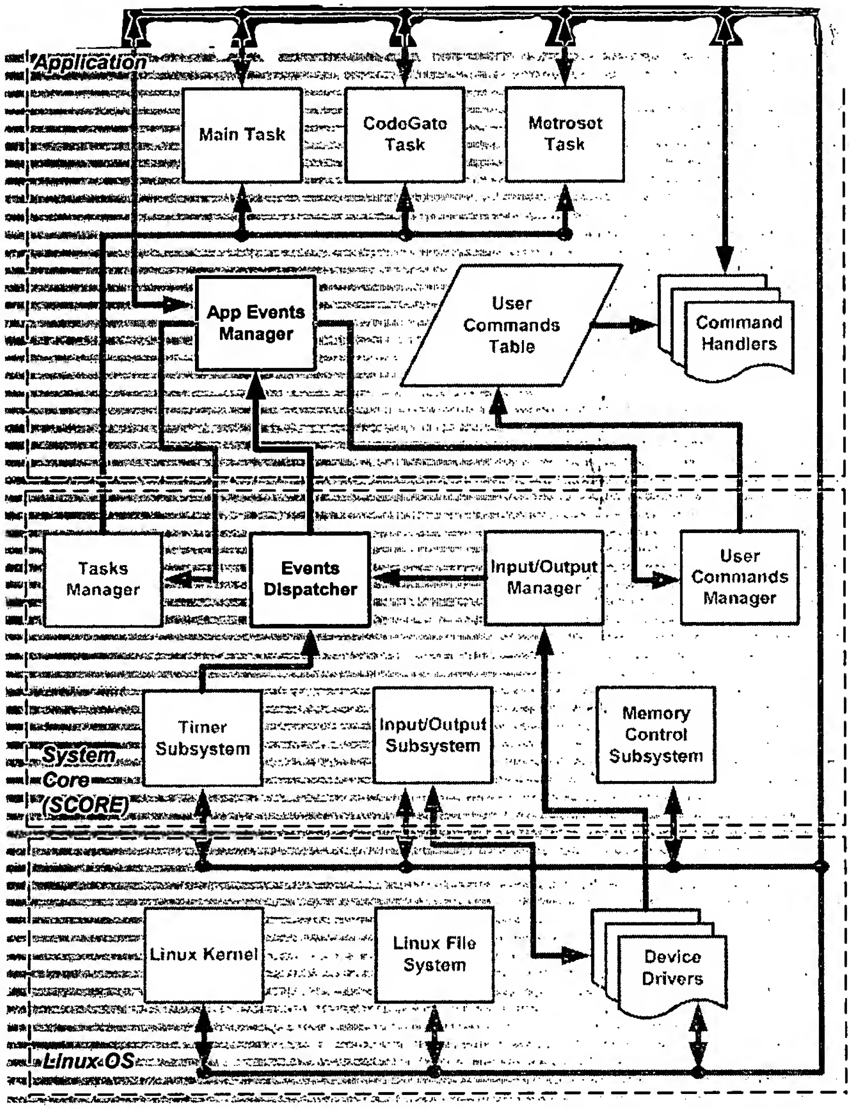
Example of Flow of Events



- The Input/Output Manager posts the SCORE_OBJECT_DETECT_ON event

FIG. 13B

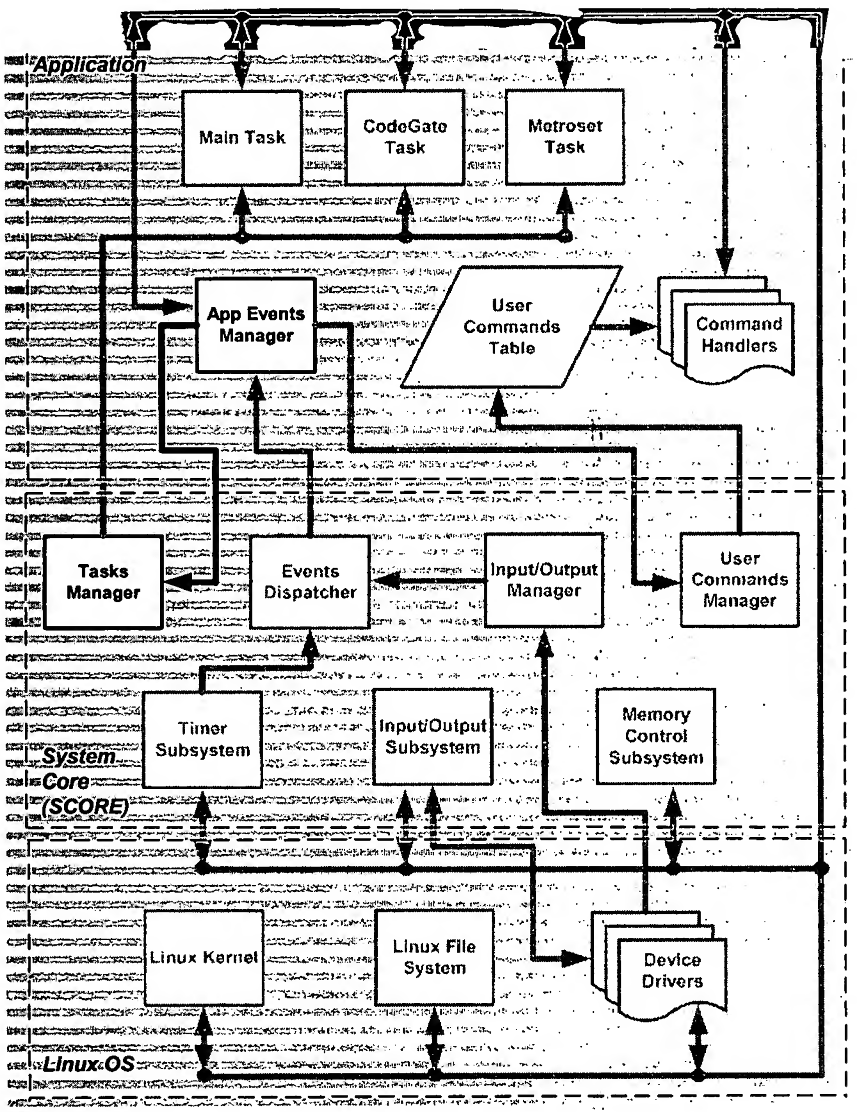
Example of Flow of Events

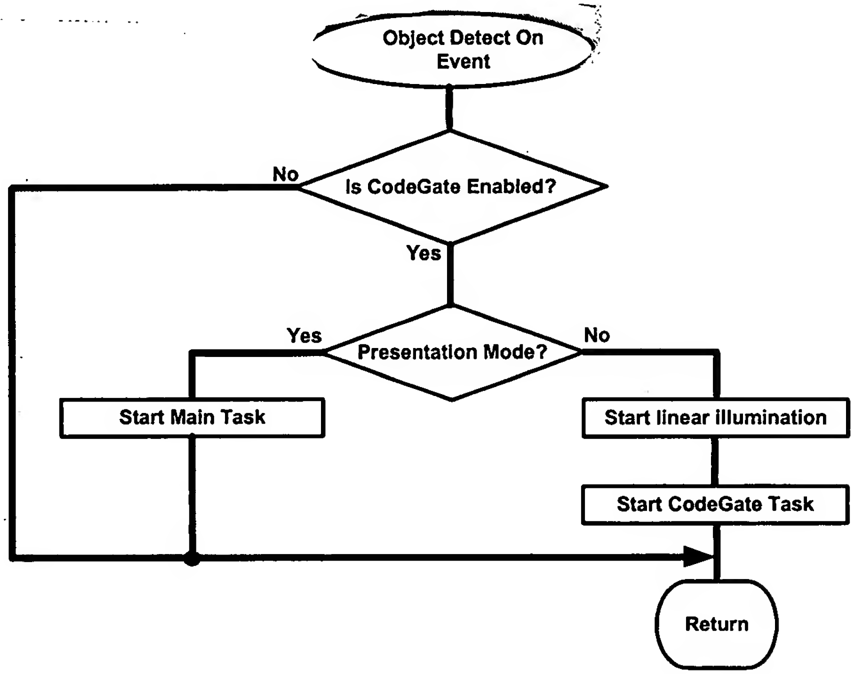


- **The Events Dispatcher passes the SCORE_OBJECT_DETECT_ON event to the application**

FIG. 13C

Example of Flow of Events

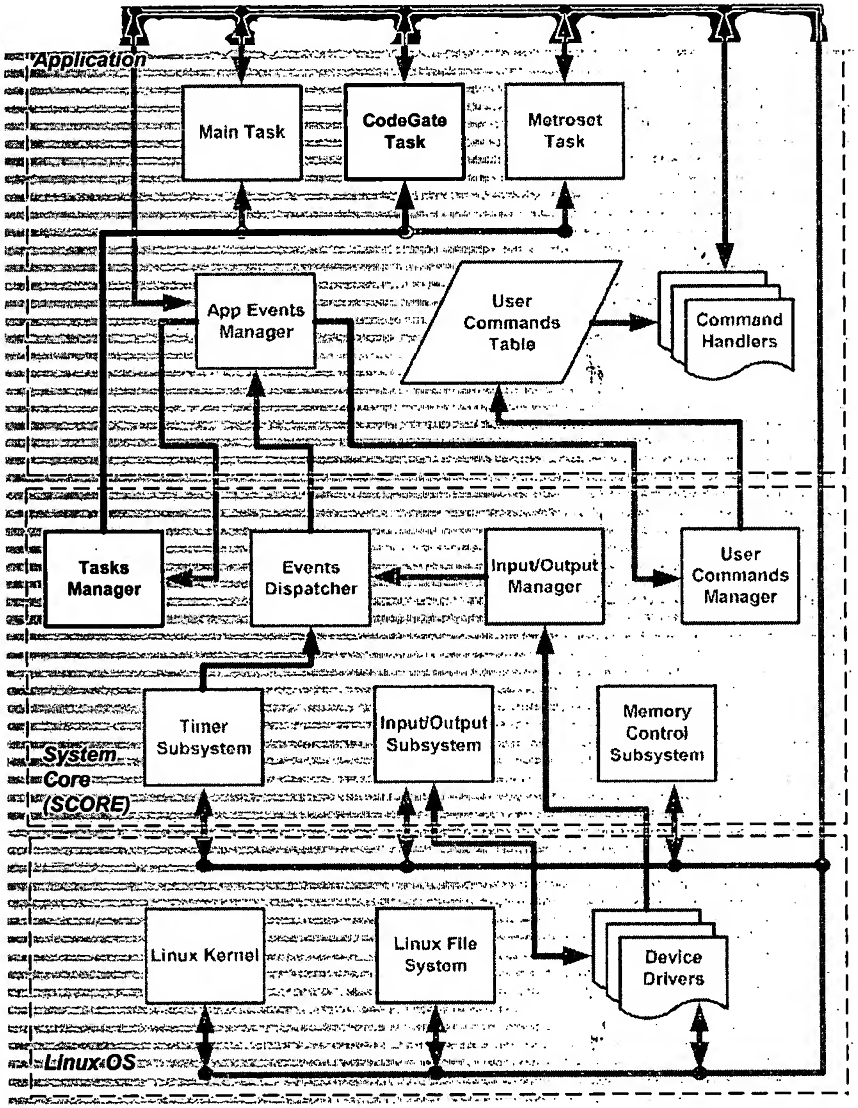




- The SCORE_OBJECT_DETECT_ON event handling routine starts linear illumination and executes the CodeGate Task

FIG. 13D

Example of Flow of Events



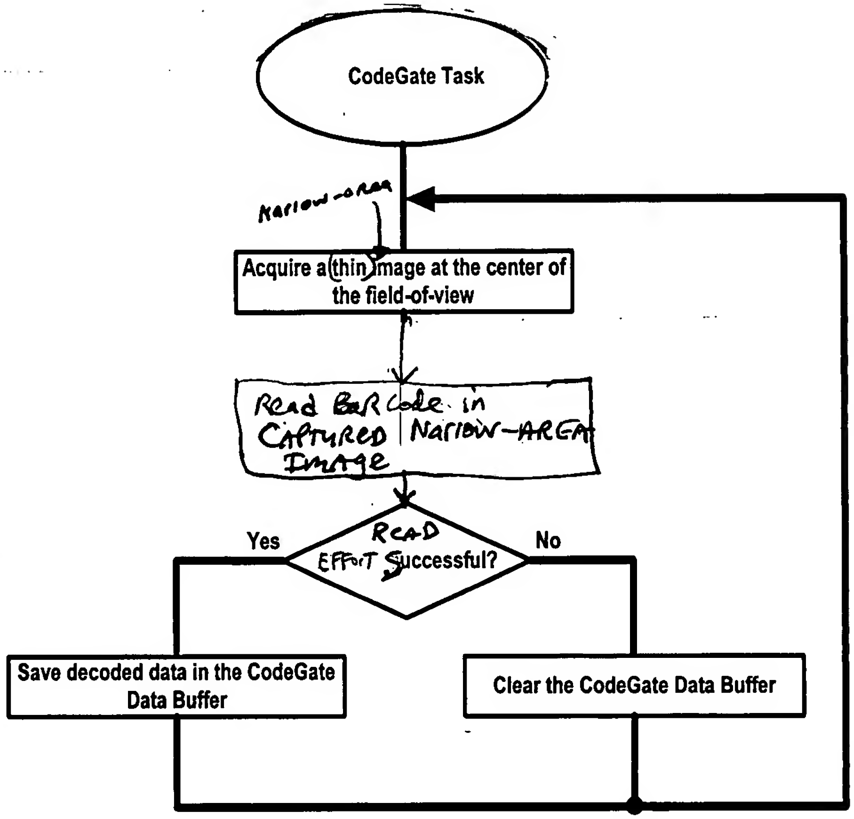
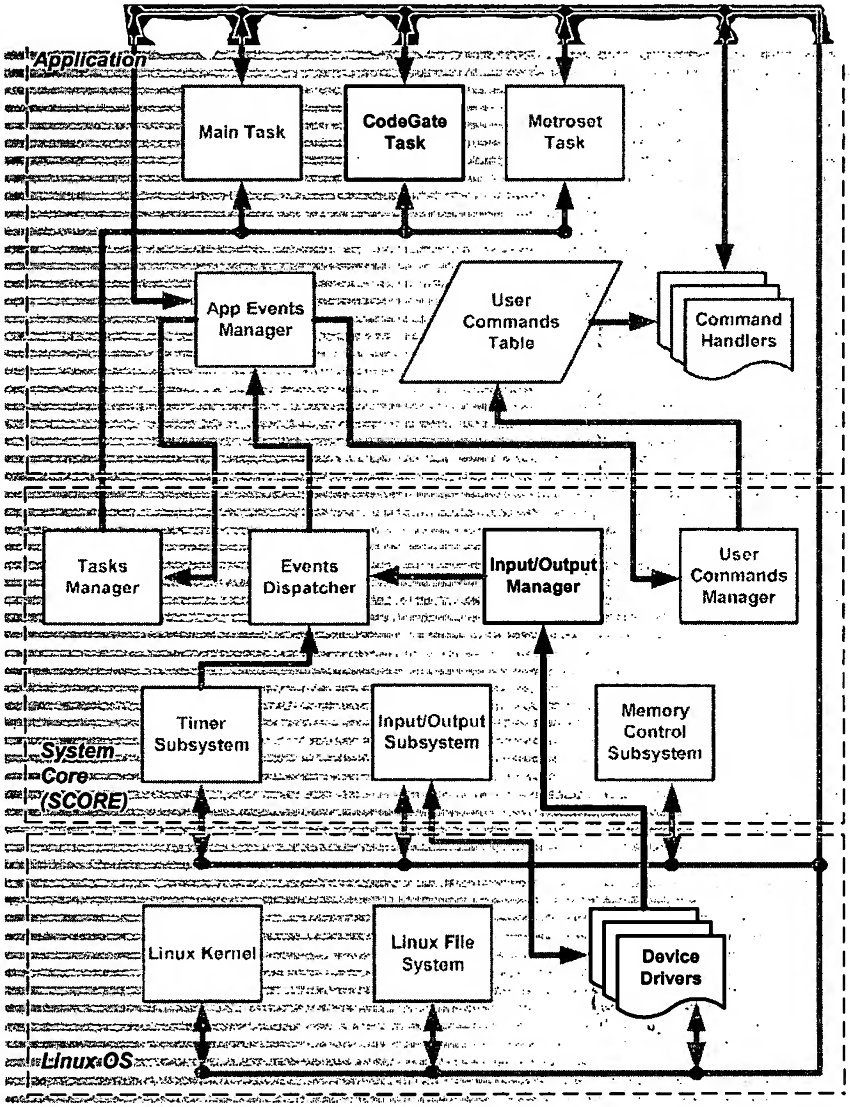

CodeGate Task
Narrow-area
Acquire a thin image at the center of the field-of-view
Read BaR Code in CAPTURED Narrow-AREA IMAGE
Read EFFoT Successful?
Yes
No
Save decoded data in the CodeGate Data Buffer
Clear the CodeGate Data Buffer

FIG. 13E

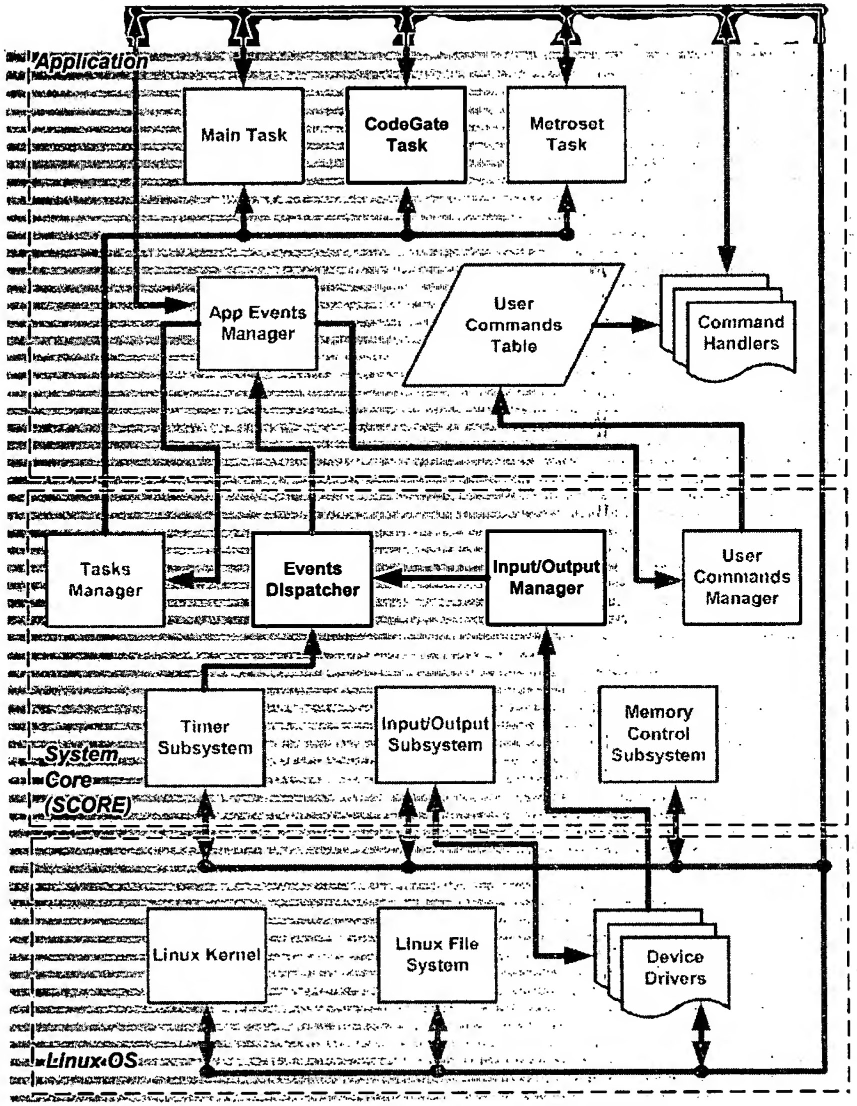
Example of Flow of Events



- User pulls the trigger
- The trigger device driver wakes up the Input/Output Manager

FIG. 13F

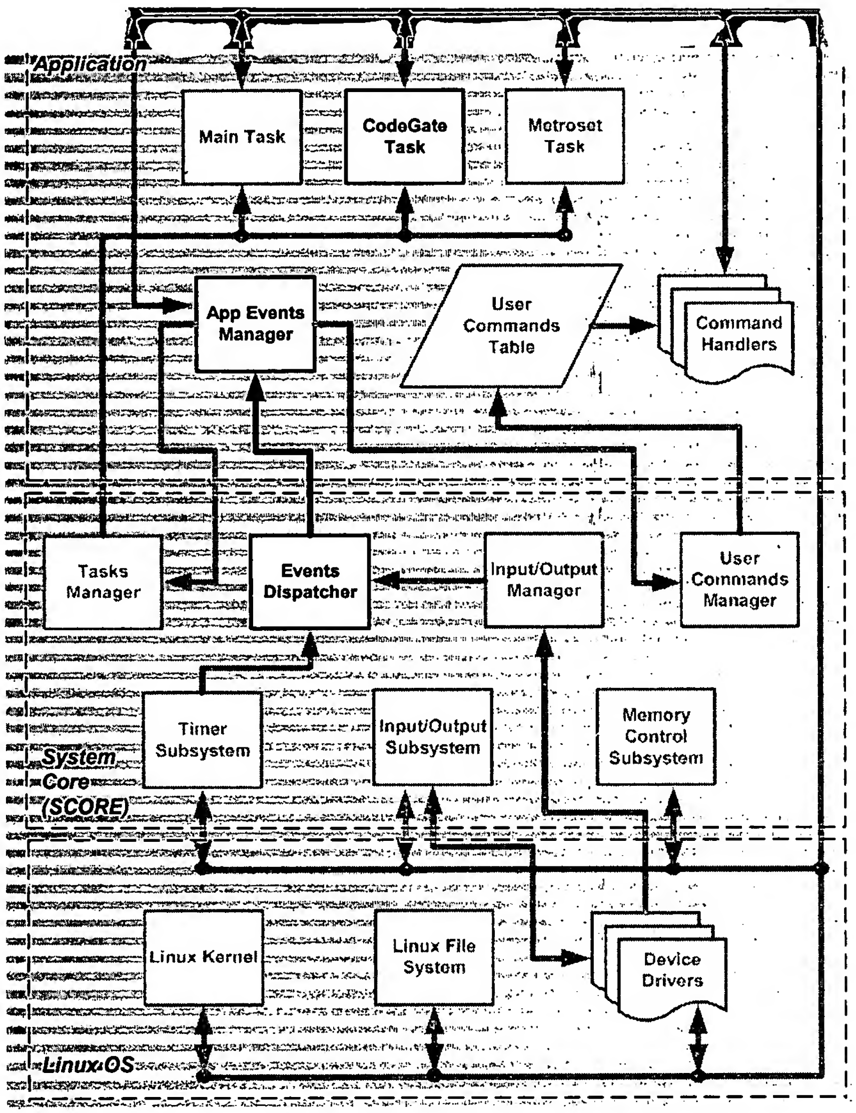
Example of Flow of Events



- **The Input/Output Manager posts the SCORE_TRIGGER_ON event**

FIG. 13G

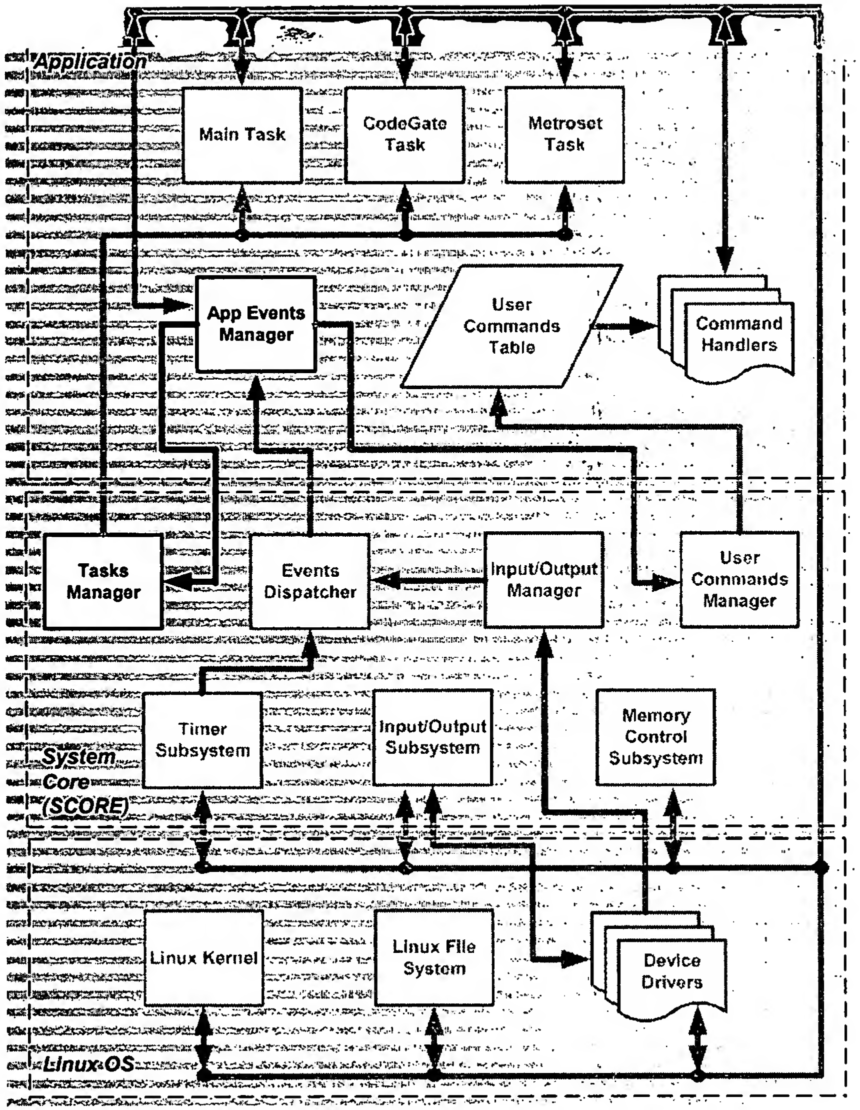
Example of Flow of Events

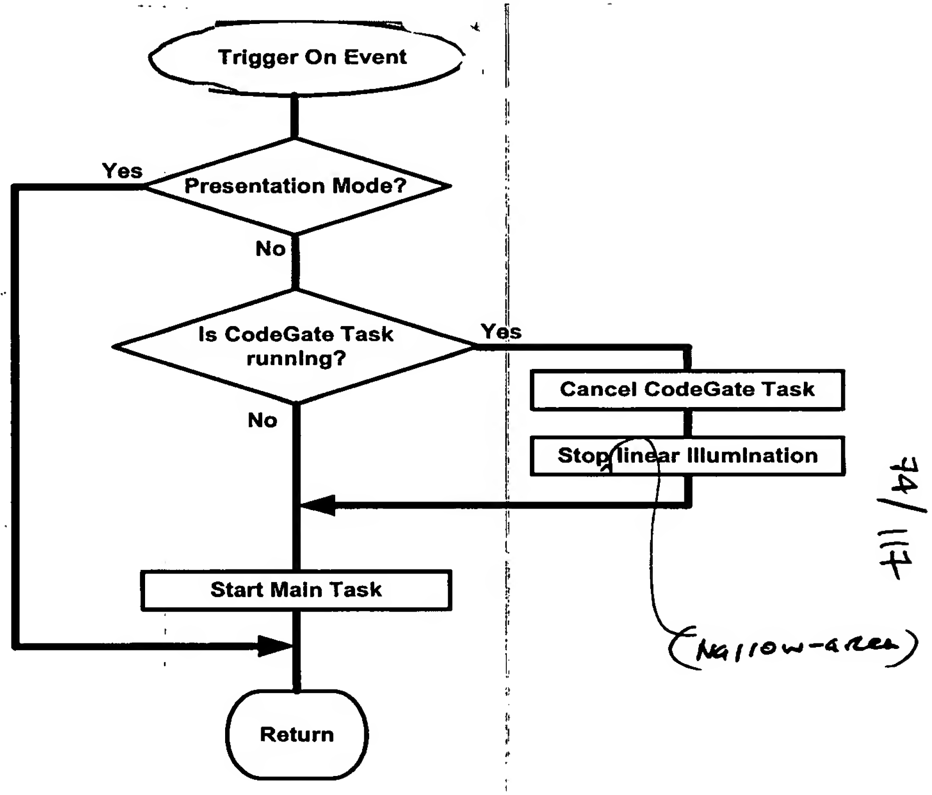


- **The Events Dispatcher passes the SCORE_TRIGGER_ON event to the application**

FIG. 13H

Example of Flow of Events





- The SCORE_TRIGGER_ON event handling routine stops linear illumination, cancels the CodeGate Task, and executes the Main Task

FIG 13I

Example of Flow of Events

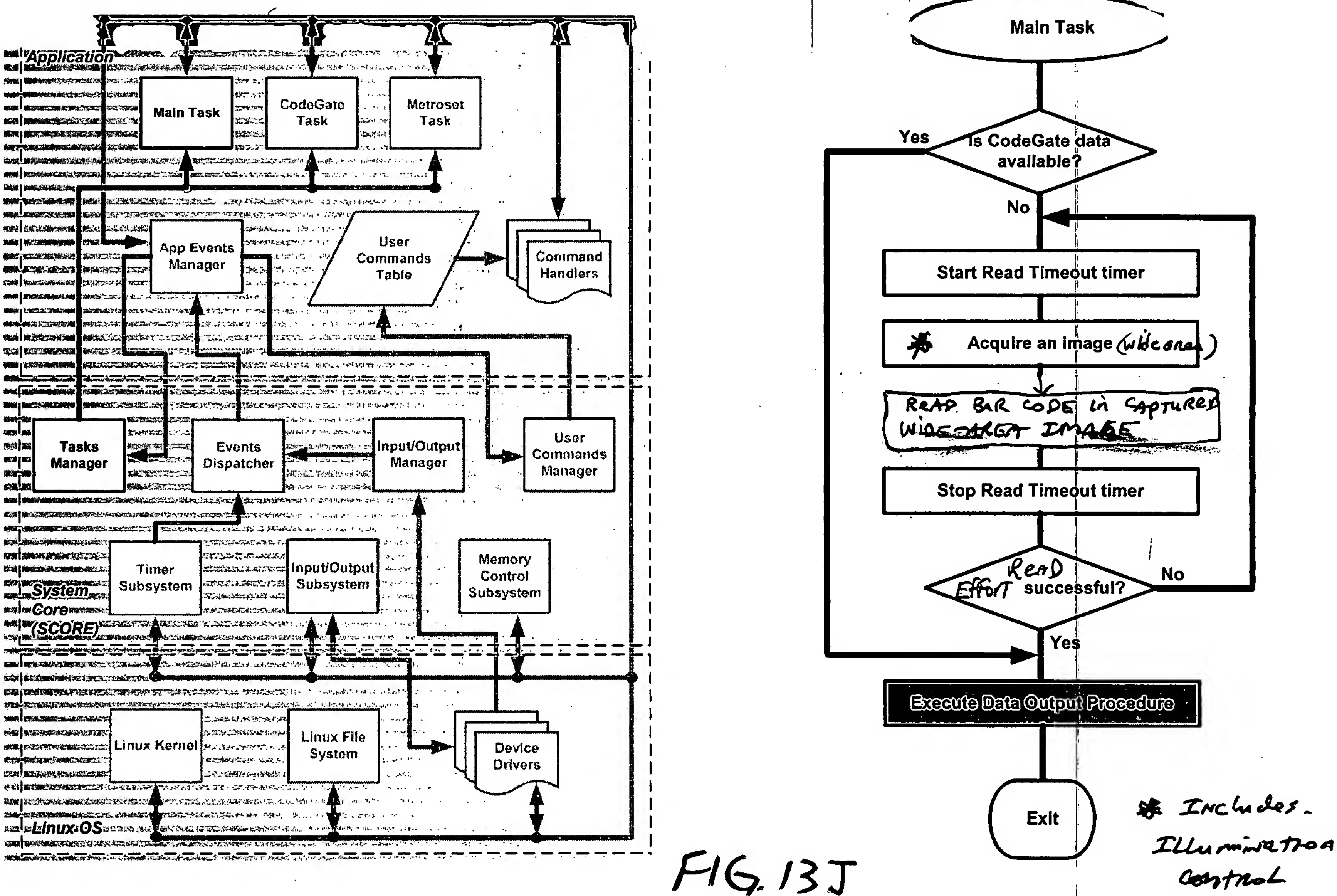


FIG. 13J

* Includes Illumination control

Example of Flow of Events

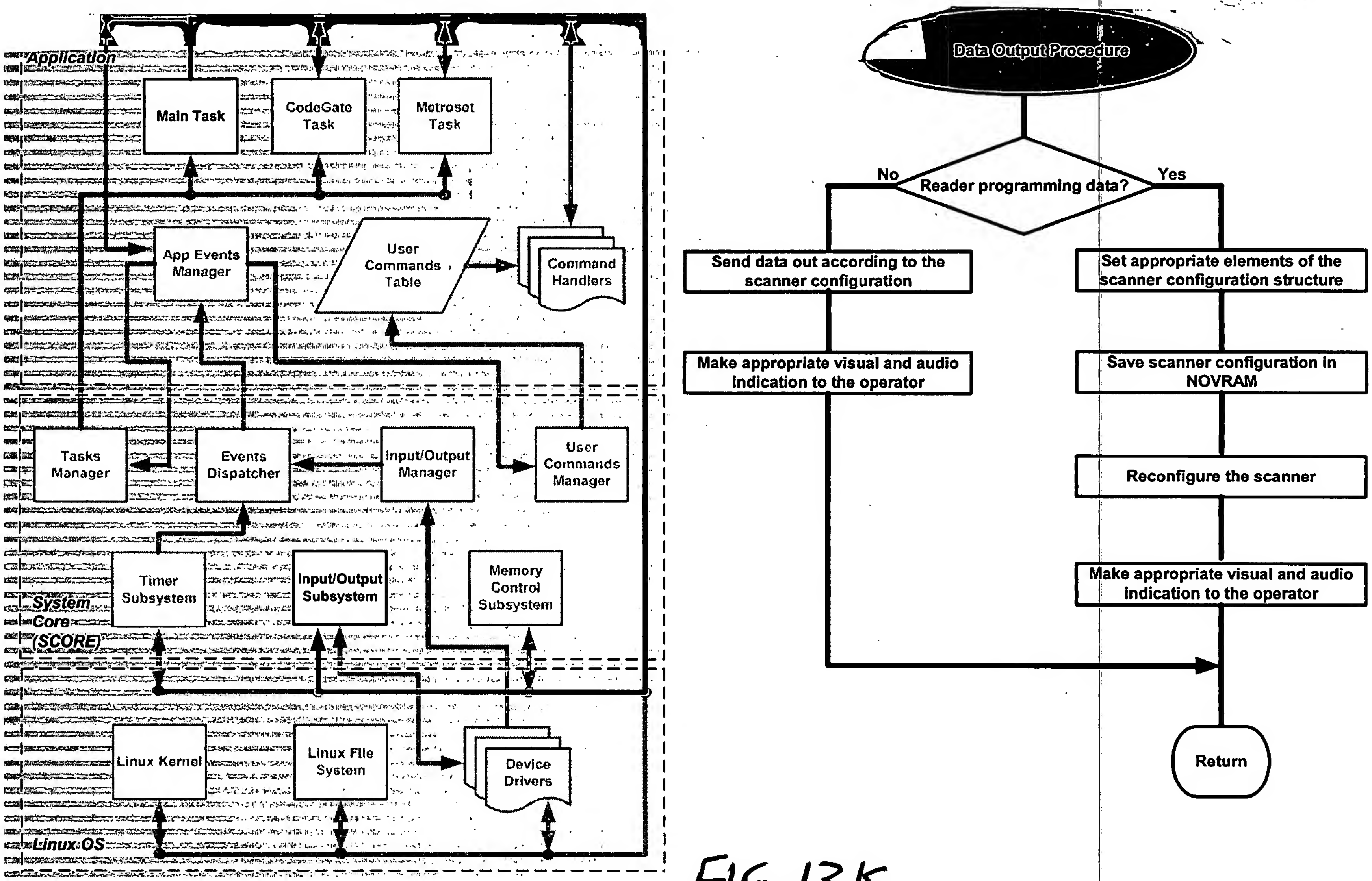
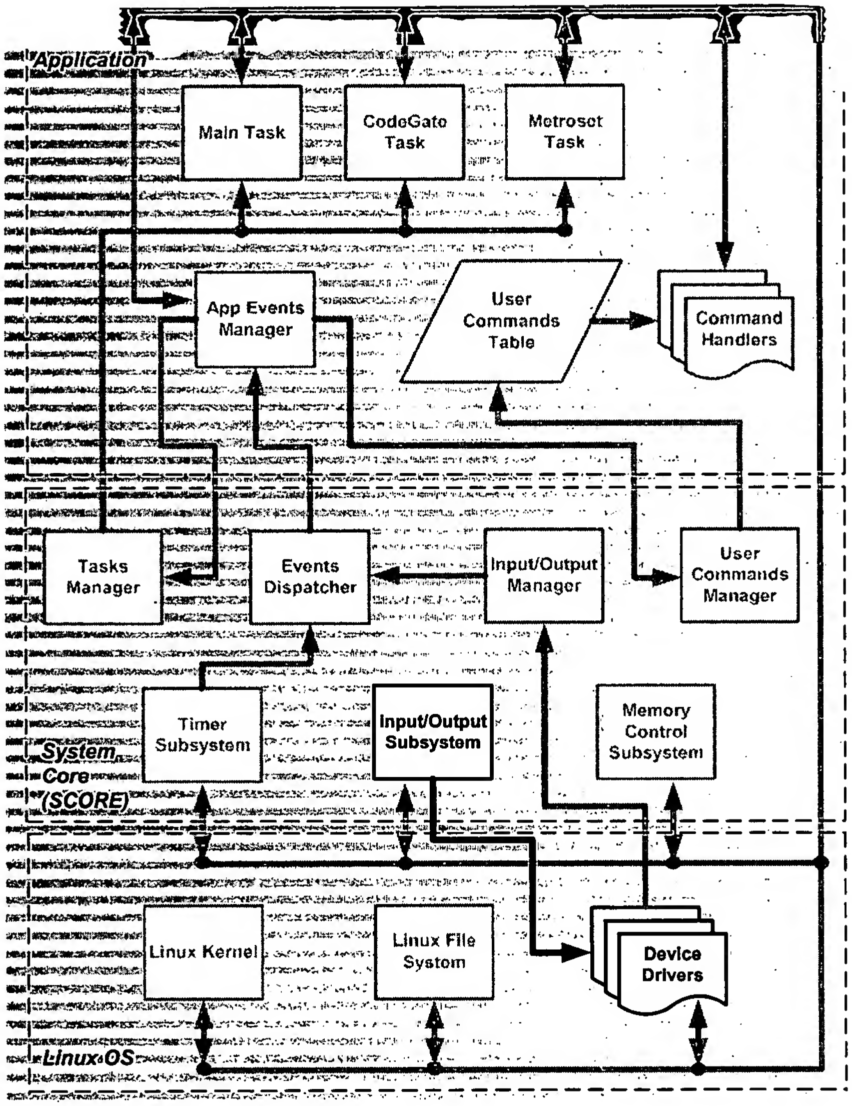


FIG. 13K

Example of Flow of Events



- The decoded data is sent to the user

FIG 13L

FIG. 13M

METHOD OF ILLUMINATING OBJECTS WITHOUT SPECULAR REFLECTION

STEP A: USE THE AUTOMATIC LIGHT EXPOSURE MEASUREMENT AND CONTROL SUBSYSTEM TO MEASURE THE LIGHT LEVEL TO WHICH THE CMOS IMAGE SENSING ARRAY IS EXPOSED.

STEP B: USE THE AUTOMATIC IR-BASED OBJECT PRESENCE AND RANGE DETECTION SUBSYSTEM TO MEASURE THE PRESENCE AND RANGE OF THE OBJECT IN EITHER THE NEAR OR FAR FIELD PORTION OF THE FIELD OF VIEW (FOV) OF THE SYSTEM.

STEP C: USE THE DETECTED RANGE AND THE MEASURED LIGHT EXPOSURE LEVEL TO DRIVE BOTH THE UPPER AND LOWER LED SUBARRAYS ASSOCIATED WITH EITHER THE NEAR OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

STEP D: CAPTURE A WIDE-AREA IMAGE AT THE CMOS IMAGE SENSING ARRAY USING THE ILLUMINATION FIELD PRODUCED DURING STEP C.

STEP E: RAPIDLY PROCESS THE CAPTURED WIDE-AREA IMAGE DURING STEP D TO DETECT THE OCCURANCE OF HIGH SPATIAL-INTENSITY LEVELS IN THE CAPTURED WIDE-AREA IMAGE, INDICATIVE OF A SPECULAR REFLECTION CONDITION.

STEP F:

IF A SPECULAR REFLECTION CONDITION IS DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN DRIVE ONLY THE UPPER LED SUBARRAY ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

IF A SPECULAR REFLECTION CONDITION IS NOT DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN USE THE DETECTED RANGE AND THE MEASURED LIGHT EXPOSURE LEVEL TO DRIVE BOTH THE UPPER AND LOWER LED SUBARRAYS ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY,.

FIG. 13M1

STEP G: CAPTURE A WIDE-AREA IMAGE AT THE CMOS IMAGE SENSING ARRAY USING THE ILLUMINATION FIELD PRODUCED DURING STEP F.

~~STEP H: RAPIDLY PROCESS THE CAPTURED WIDE-AREA IMAGE DURING~~ STEP G TO DETECT THE OCCURANCE OF HIGH SPATIAL-INTENSITY LEVELS IN THE CAPTURED WIDE-AREA IMAGE, INDICATIVE OF A SPECULAR REFLECTION CONDITION.

STEP I:

IF A SPECULAR REFLECTION CONDITION IS STILL DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN DRIVE THE OTHER LED SUBARRAY ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

IF A SPECULAR REFLECTION CONDITION IS NOT DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN DRIVE USE THE DETECTED RANGE AND THE MEASURED LIGHT EXPOSURE LEVEL TO DRIVE THE SAME LED SUBARRAY (AS IN STEP C) ASSOCIATED WITH EITHER THE NEAR FIELD OR FAR FIELD WIDE AREA ILLUMINATION ARRAY.

STEP J: CAPTURE A WIDE-AREA IMAGE AT THE CMOS IMAGE SENSING ARRAY USING THE ILLUMINATION FIELD PRODUCED DURING STEP I.

STEP K: RAPIDLY PROCESS THE CAPTURED WIDE-AREA IMAGE DURING STEP J TO Detect the ABSENCE OF HIGH SPATIAL-INTENSITY LEVELS IN THE CAPTURED WIDE-AREA IMAGE, CONFIRMING THE ELIMINATION OF THE ONCE DETECTED SPECULAR REFLECTION CONDITION.

FIG. 13M2

STEP L:

IF NO SPECULAR REFLECTION CONDITION IS DETECTED IN THE PROCESSED WIDE-AREA IMAGE AT STEP K, THEN PROCESS THE WIDE-AREA IMAGE USING MODE(S)SELECTED FOR THE MULTI-MODE IMAGE-PROCESSING BAR CODE READING SUBSYSTEM.

IF A SPECULAR REFLECTION CONDITION IS STILL DETECTED IN THE PROCESSED WIDE-AREA IMAGE, THEN RETURN TO STEP A REPEAT STEPSA THROUGH K.

FIG. 13M3

Symbologies READABLE BY Multi-mode BAR CODE SYMBOL Reading Subsystem

(1) Code 128

(2) Code 39

(3) I2of5

(4) Code93

(5) Codabar

(6) UPC/EAN

(7) Telepen

(8) UK-Plessey

(9) Trioptic

(10) Matrix 2of5

(11) Airline 2of5

(12) Straight 2of5

(13) MSI-Plessey

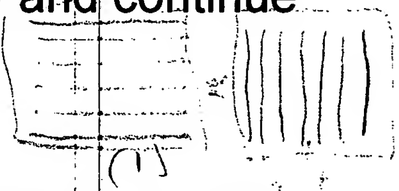
(14) Code11

(15) PDF417

FIG. 14

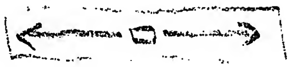
Modes of operation OF MULTI-MODE BAR Code Reading Subsystem

- Automatic – look for multiple barcodes incrementally and continue looking until entire image is processed



- Manual – look for a programmable number of barcodes starting from center of image

- NoFinder – look for one barcode in picket-fence orientation starting from center of image



- OmniScan – look for one barcode along pre-determined orientations

- ROI-specific - Method Look FOR bar code at specified Region of Interest (ROI) in captured Image.

FIG. 15

setup and cleanup flow-chart

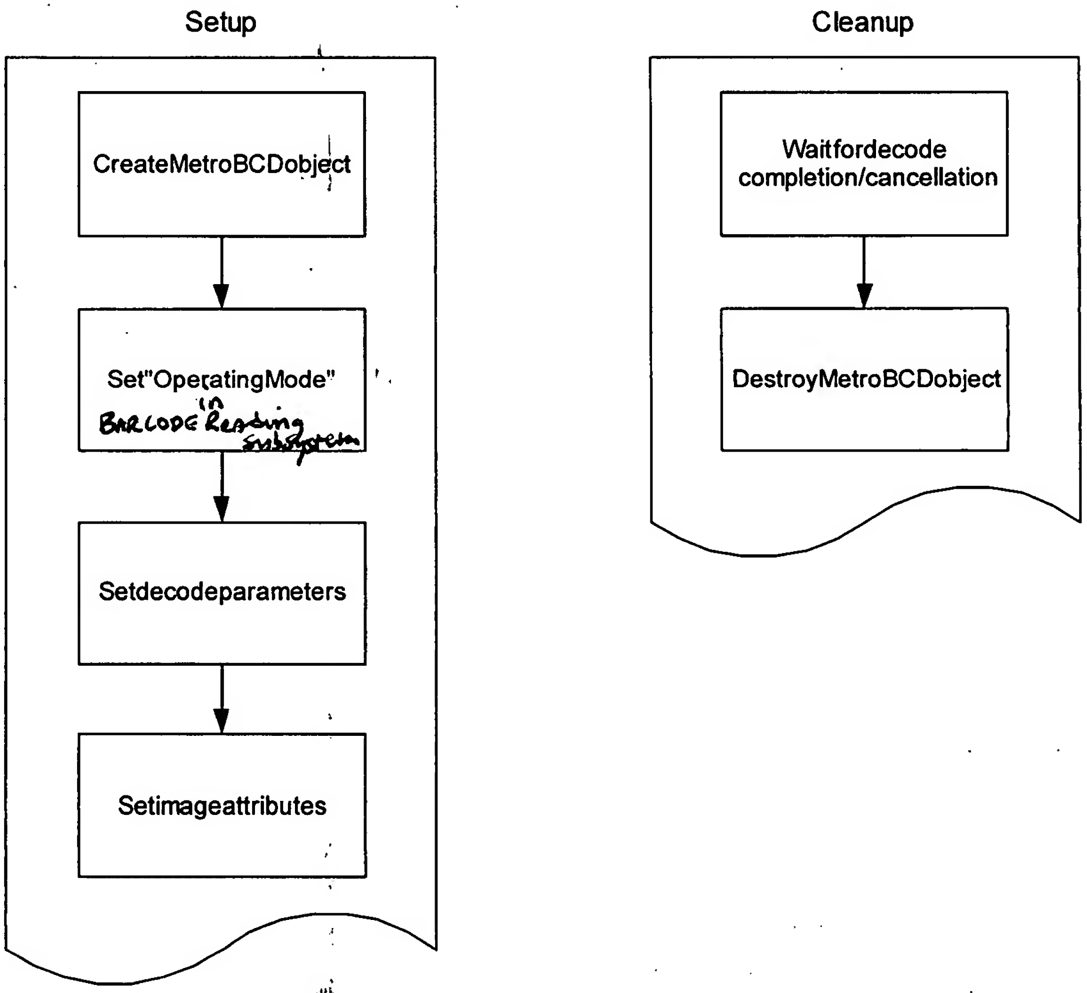
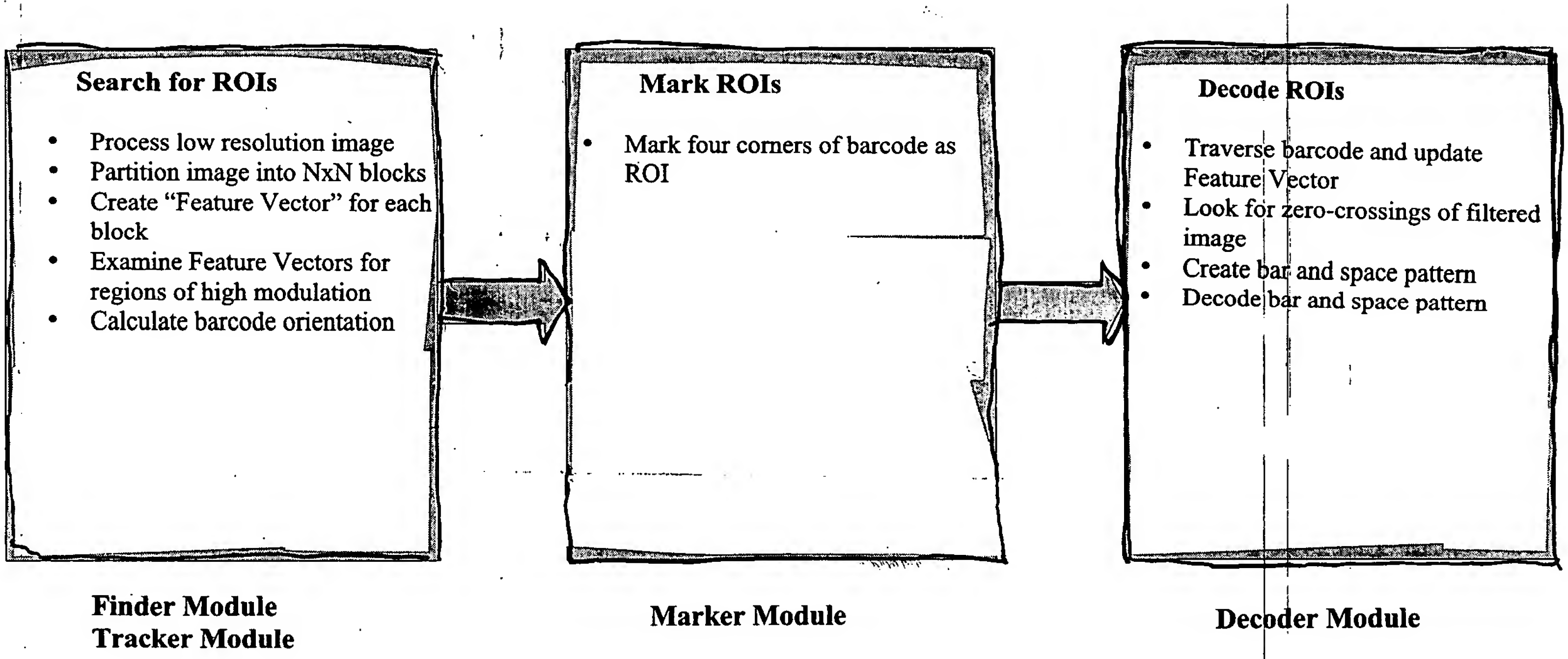


FIG. 16

Summary of Automatic mode



Search for ROIs

- Process low resolution image
- Partition image into NxN blocks
- Create "Feature Vector" for each block
- Examine Feature Vectors for regions of high modulation
- Calculate barcode orientation

Finder Module
Tracker Module

Mark ROIs

- Mark four corners of barcode as ROI

Marker Module

Decode ROIs

- Traverse barcode and update Feature Vector
- Look for zero-crossings of filtered image
- Create bar and space pattern
- Decode bar and space pattern

Decoder Module

FIG. 17A

Automatic mode flow-chart

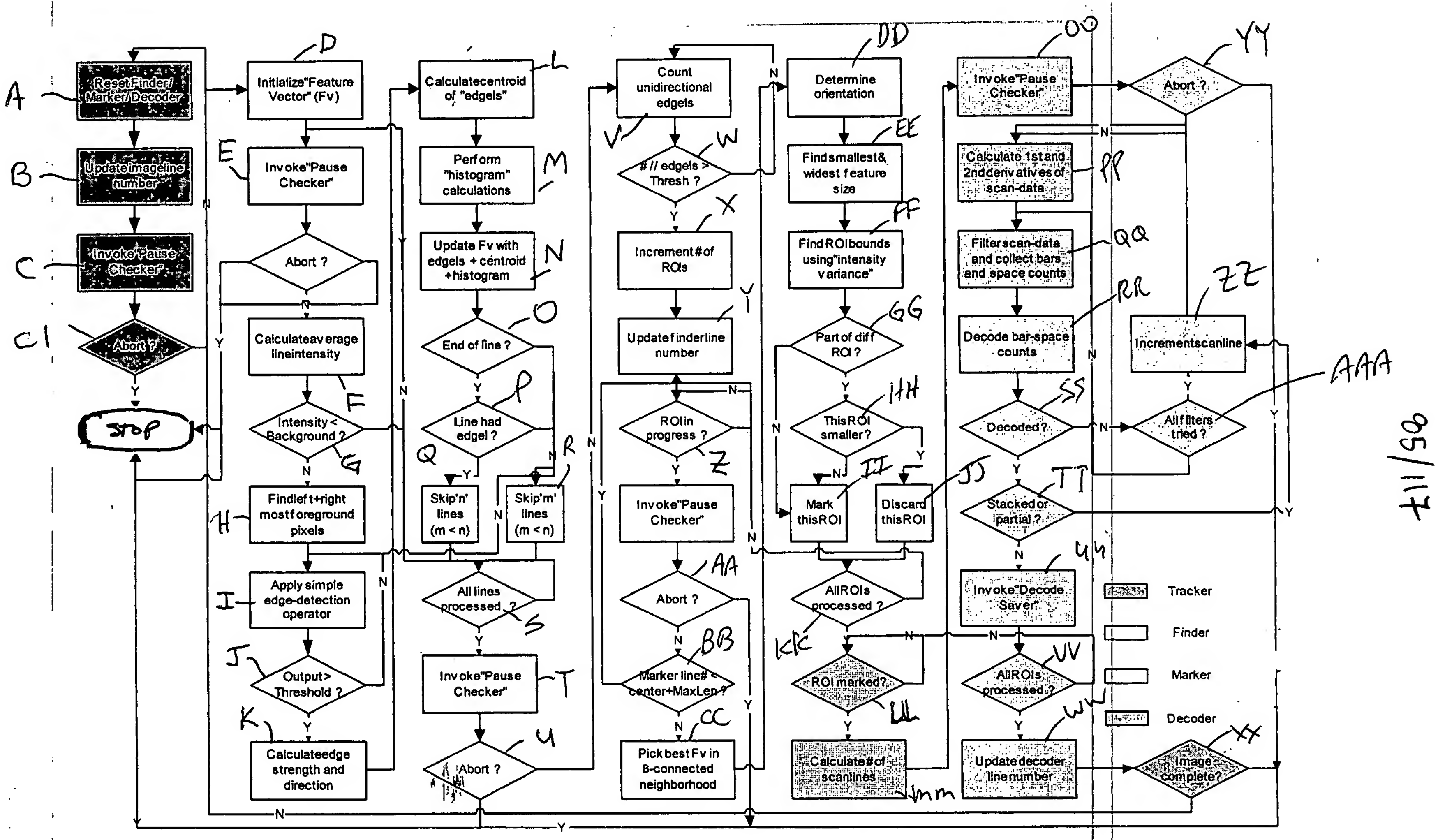
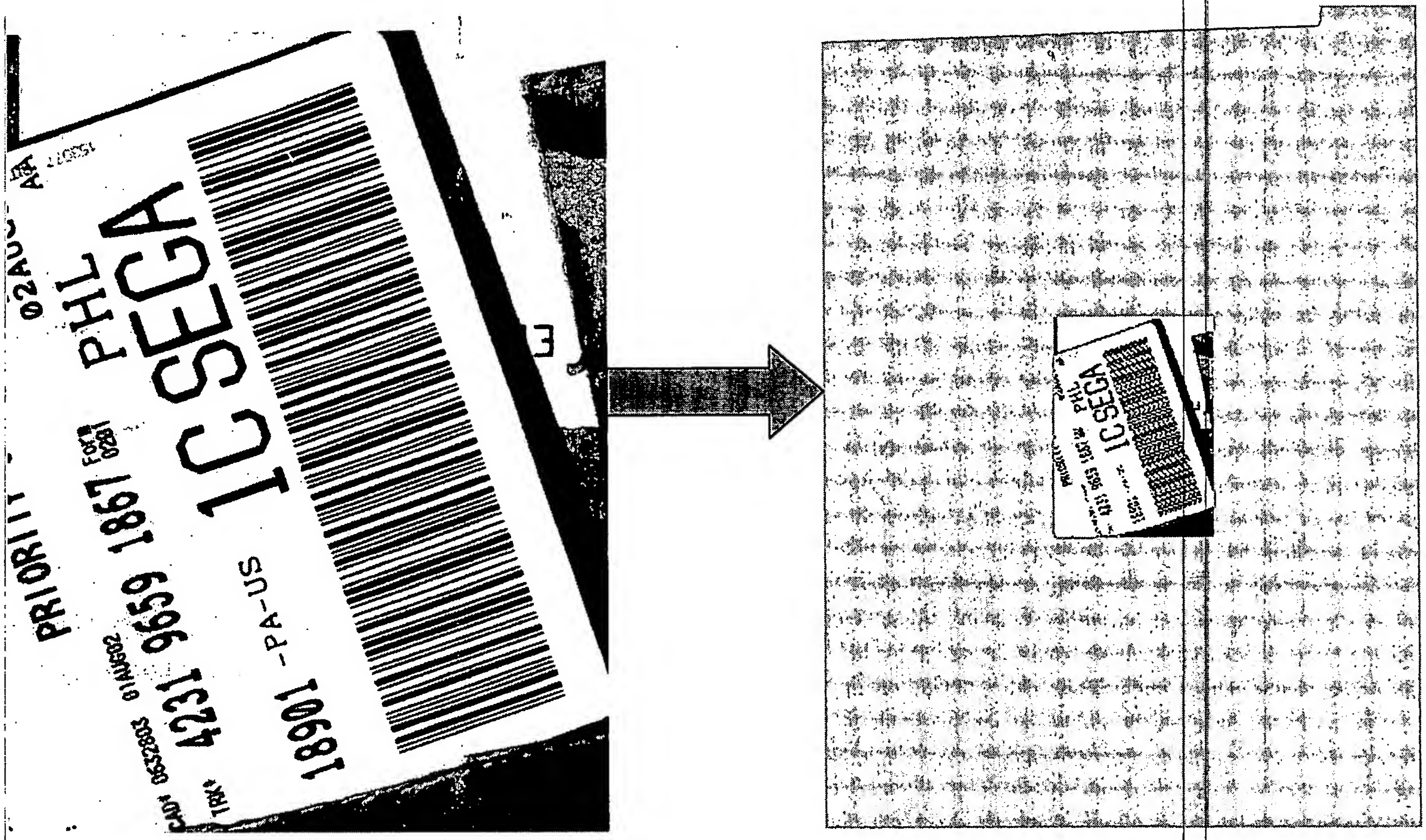


FIG. 17B

Step 1: Search for ROIs: Low resolution processing

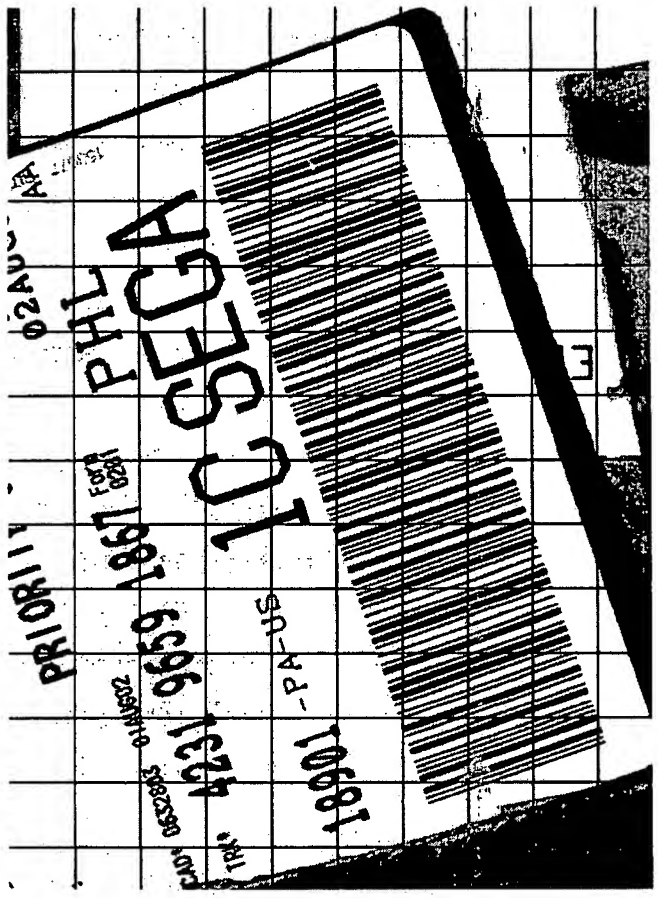


Original Image

Low-Resolution Image

FIG. 18A

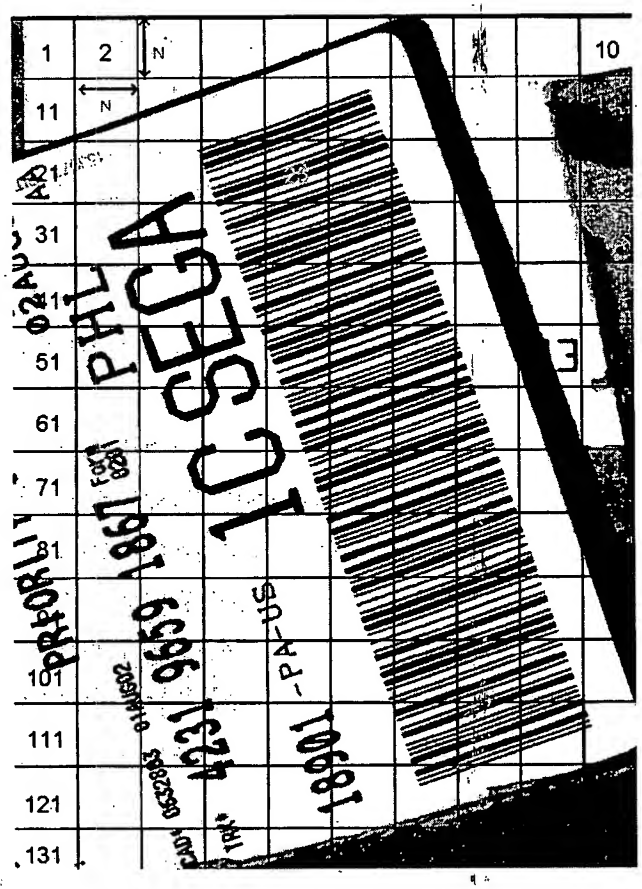
Step 2: Search for ROIs: Partition image



- Image overlaid with XY grid
- Each block formed by grids has an associated “feature vector” (Fv)
- Feature vectors are analyzed for the presence of parallel lines
- All feature vector calculations are performed on the low-resolution image

FIG. 18B

Step 3: Search for ROIs: Create feature vectors

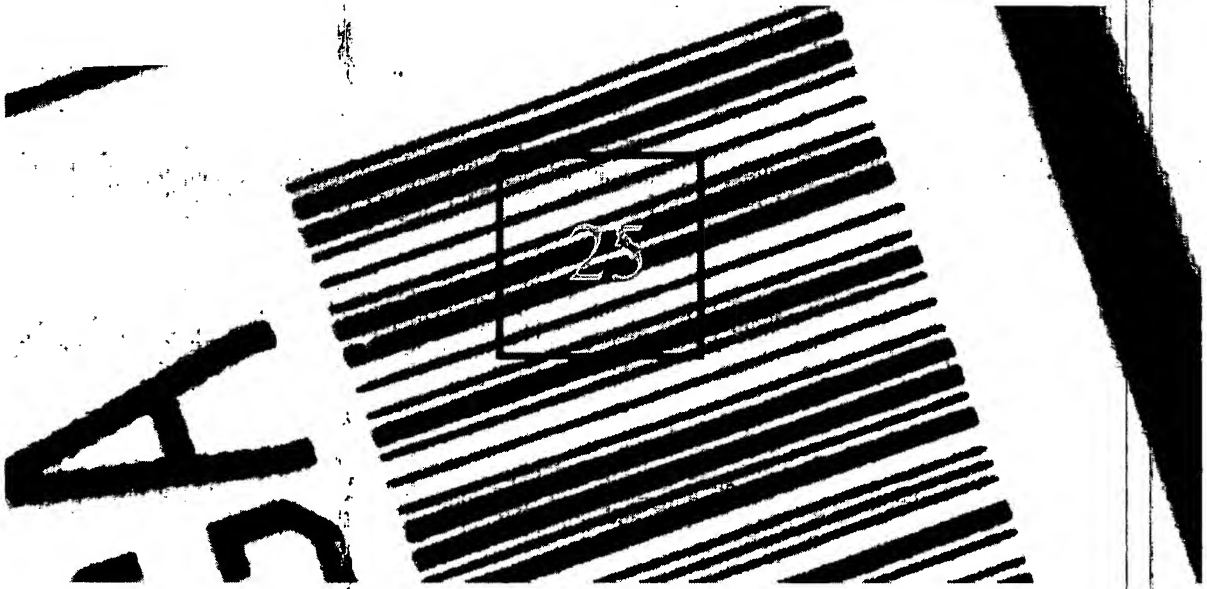


Fv =

- Gradient vectors
- Edge density
- Number of parallel edge vectors
- Centroid of edgels
- Intensity variance
- Histogram of intensities

FIG. 18C

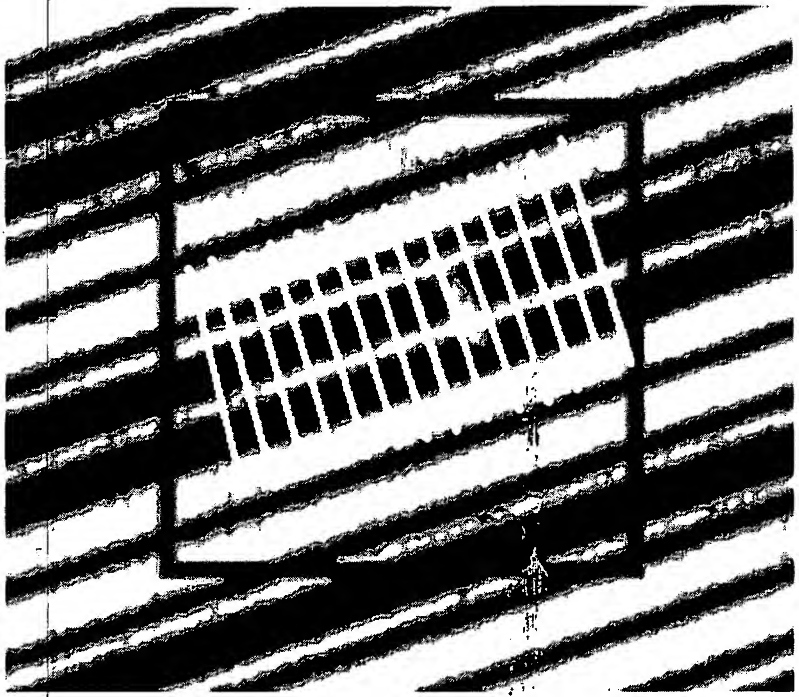
Step 4: Mark ROIs: Examine feature vectors



- High edge density
- Large number of parallel edge vectors
- Large intensity variance

FIG.18D

Step 5: Mark ROIs: Calculate barcode orientation



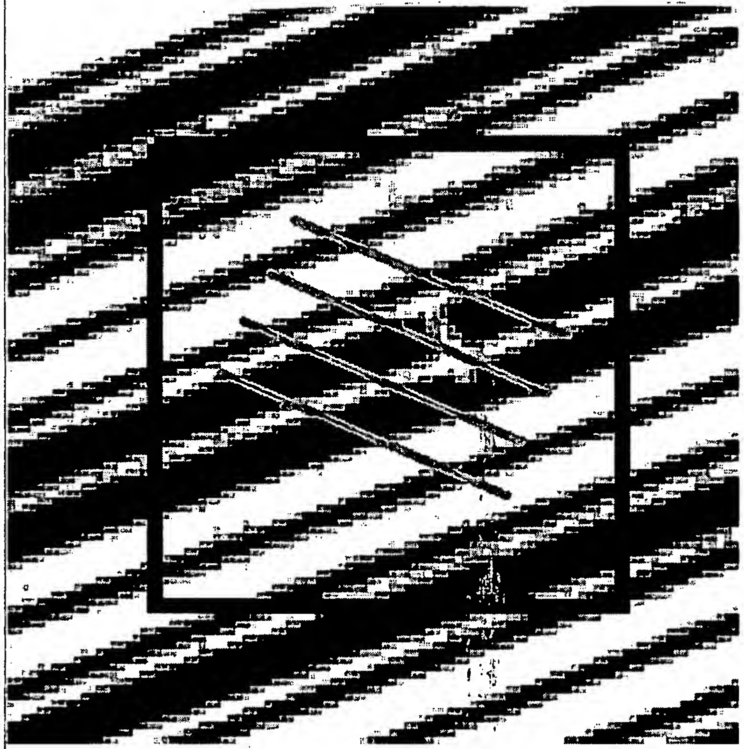
- Within each “feature vector” block the barcode is traversed (“sliced”) at different angles

- The slices are matched with each other based on “least mean square error”

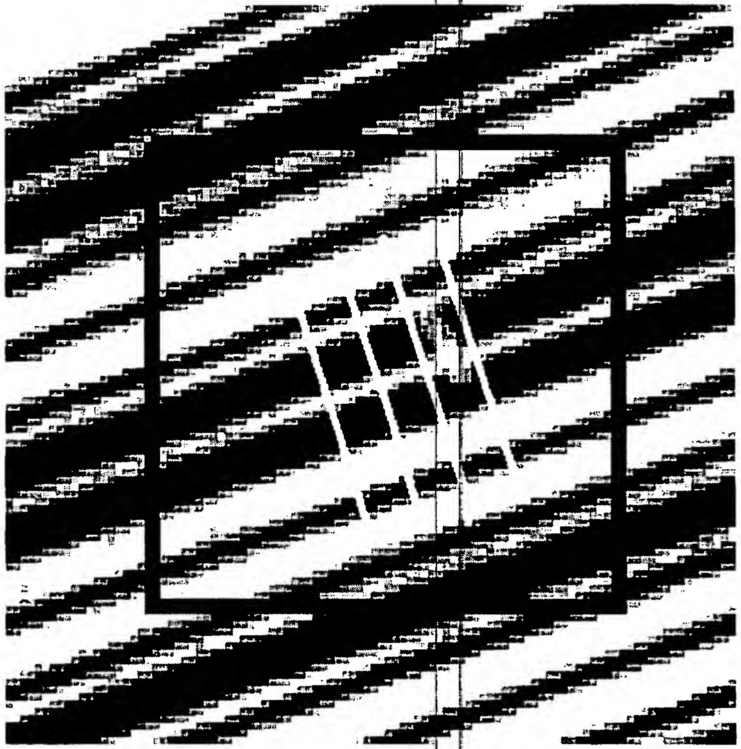
- The correct orientation is that angle that matches in a “mean square error” sense every slice of the barcode

FIG. 18E

Step 5: Mark ROIs: Calculate barcode orientation



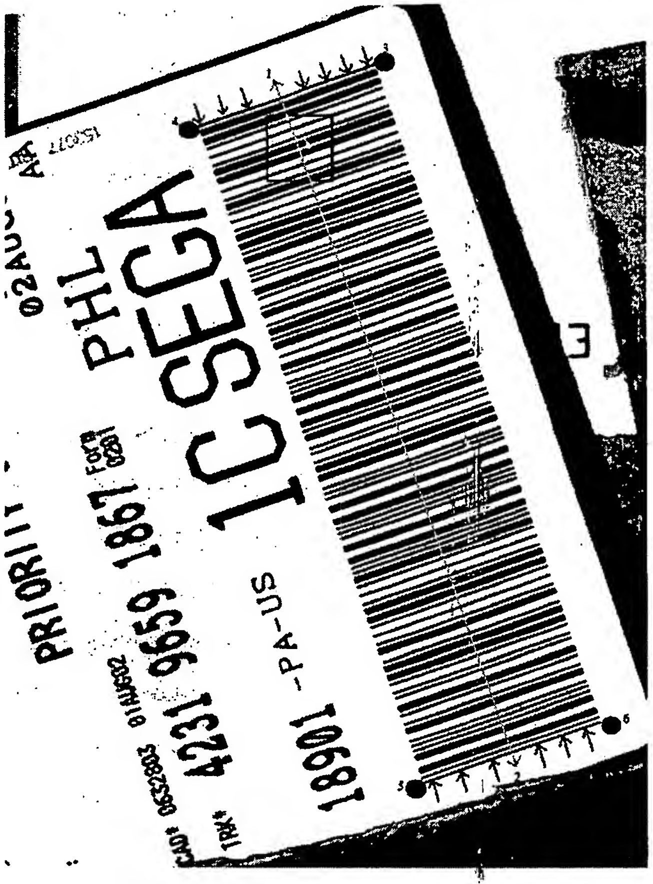
High mean square error between slices



Lowest mean square error between slices
- Correct orientation

FIG. 18F

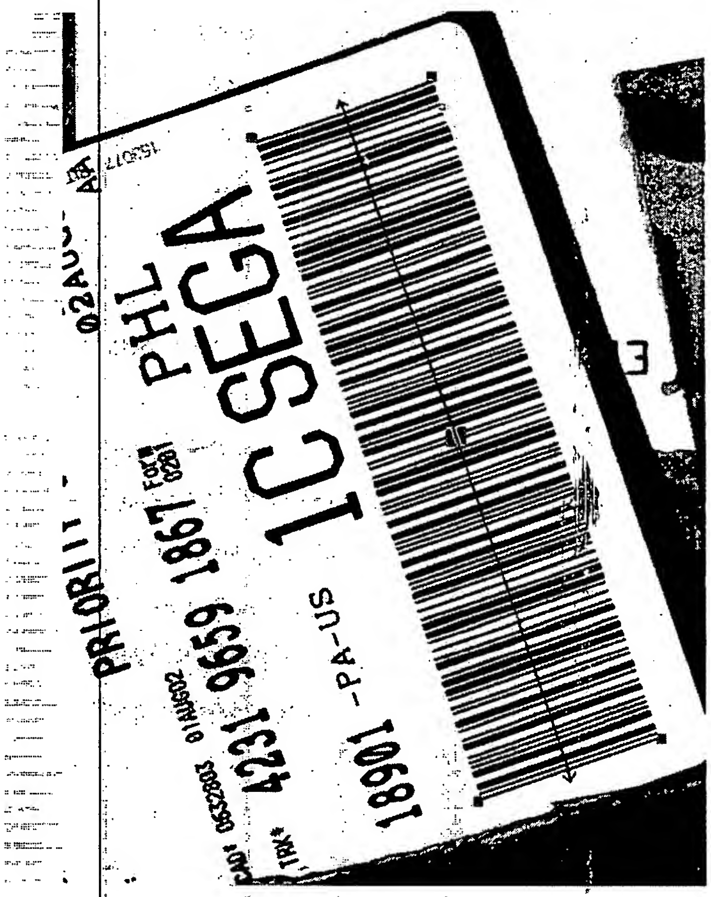
Step 6: Mark ROIs: Mark four corners of barcode

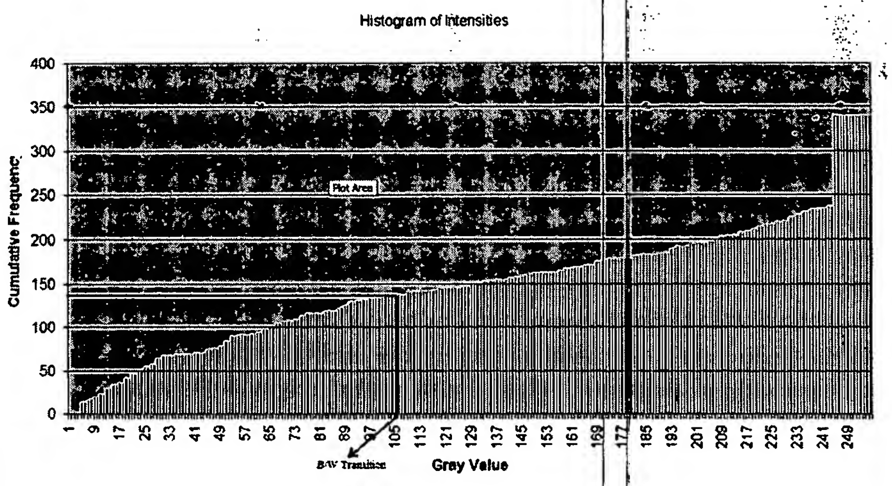


- From here on all operations are performed on the full-resolution image
- Barcode is traversed in either direction starting from center of block
- Using intensity variance the extent of modulation is detected (1 & 2)
- Starting from 1 & 2 and moving perpendicular to barcode orientation the four corners are determined (3, 4, 5, 6)
- 3, 4, 5, 6 define the ROI

FIG. 18G

Step 7: Decode ROIs: Update feature vectors

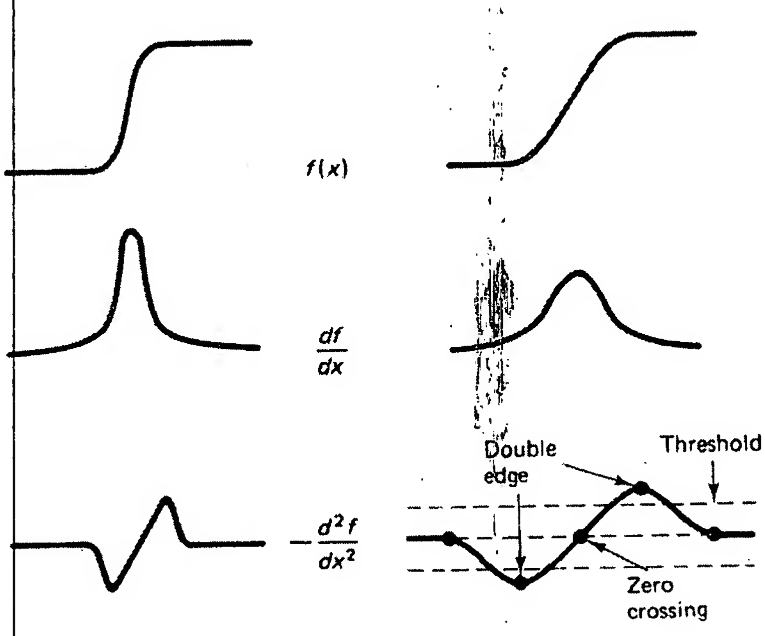




- Histogram component of Fv is updated while traversing barcode
- Estimate of Black-to-White transition is calculated
- Estimate of narrow & wide elements are calculated

FIG. 18H

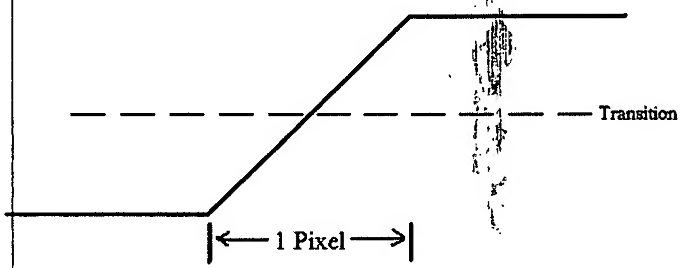
Step 8: Decode ROIs: Look for zero-crossings



- Barcode image is median filtered in a direction perpendicular to barcode orientation
- Second derivative zero-crossings define edge transitions
- Zero-crossing data used only for detecting the edge transitions
- B/W transition estimates put upper and lower bounds to bar and space gray levels

FIG. 18I

Step 9: Decode ROIs: Create bar and space pattern



- Edge transition is modeled as a ramp
- Edge transition is assumed to be 1-pixel wide
- Edge transition location is determined at the sub-pixel level
- Bar and space counts are gathered using edge transition data

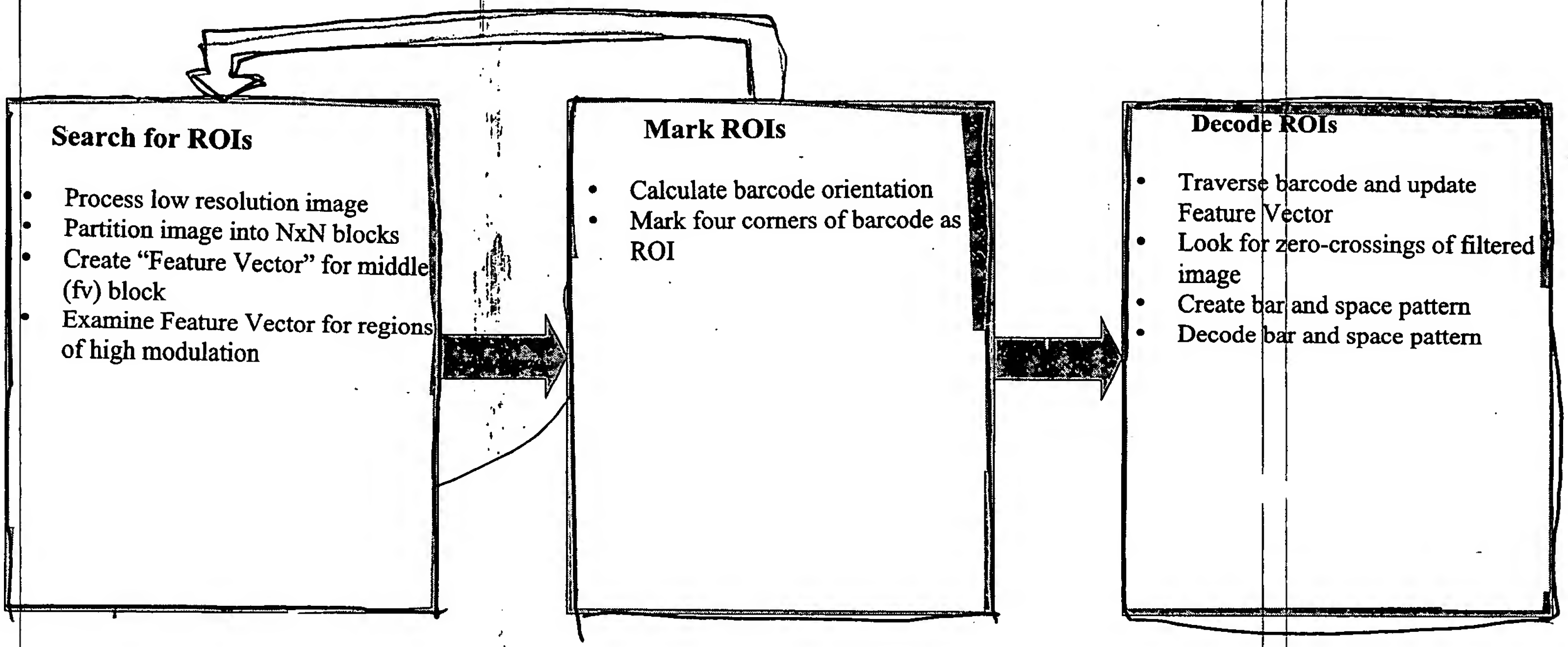
FIG. 18J

Step 10: Decode ROIs: Decode bar and space pattern

- Bar and space data framed with “borders”
- Bar and space data decoded using existing Metrologic laser-scanner algorithms

FIG. 18K

Summary of Manual mode



Search for ROIs

- Process low resolution image
- Partition image into NxN blocks
- Create "Feature Vector" for middle (fv) block
- Examine Feature Vector for regions of high modulation

Mark ROIs

- Calculate barcode orientation
- Mark four corners of barcode as ROI

Decode ROIs

- Traverse barcode and update Feature Vector
- Look for zero-crossings of filtered image
- Create bar and space pattern
- Decode bar and space pattern

FIG. 19A

Manual mode flow-chart

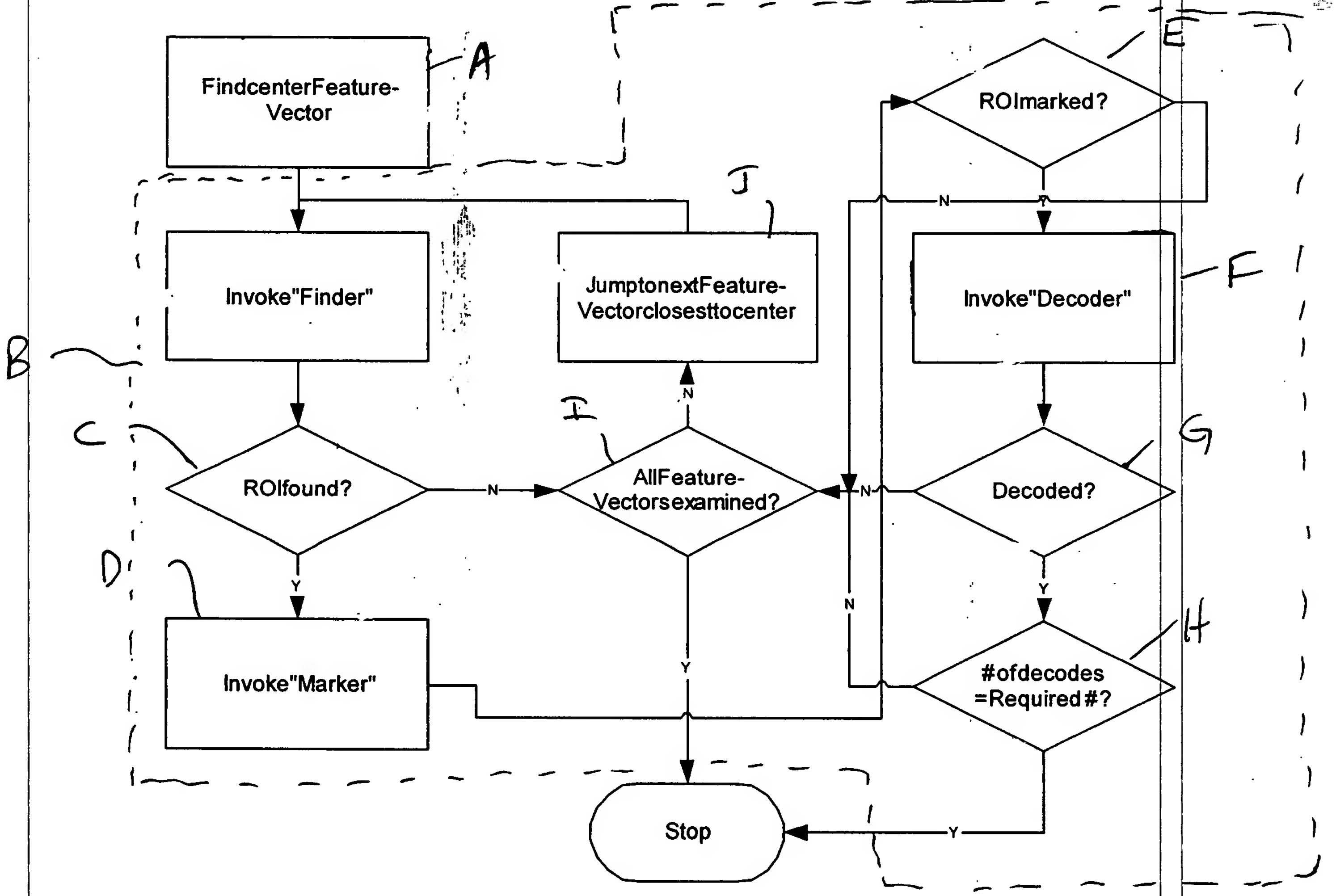
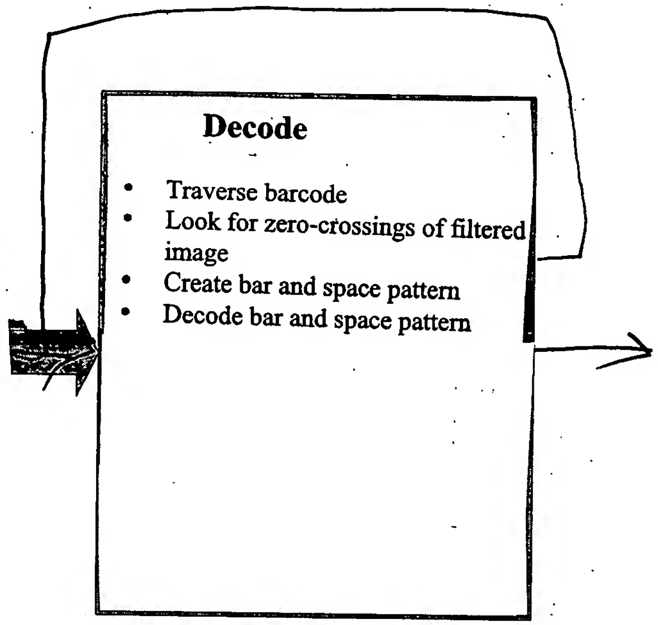


FIG. 19B

Summary of No Finder mode



Decode

- Traverse barcode
- Look for zero-crossings of filtered image
- Create bar and space pattern
- Decode bar and space pattern

- No Finder
- No Marker

FIG. 20A

NoFinder mode flow-chart

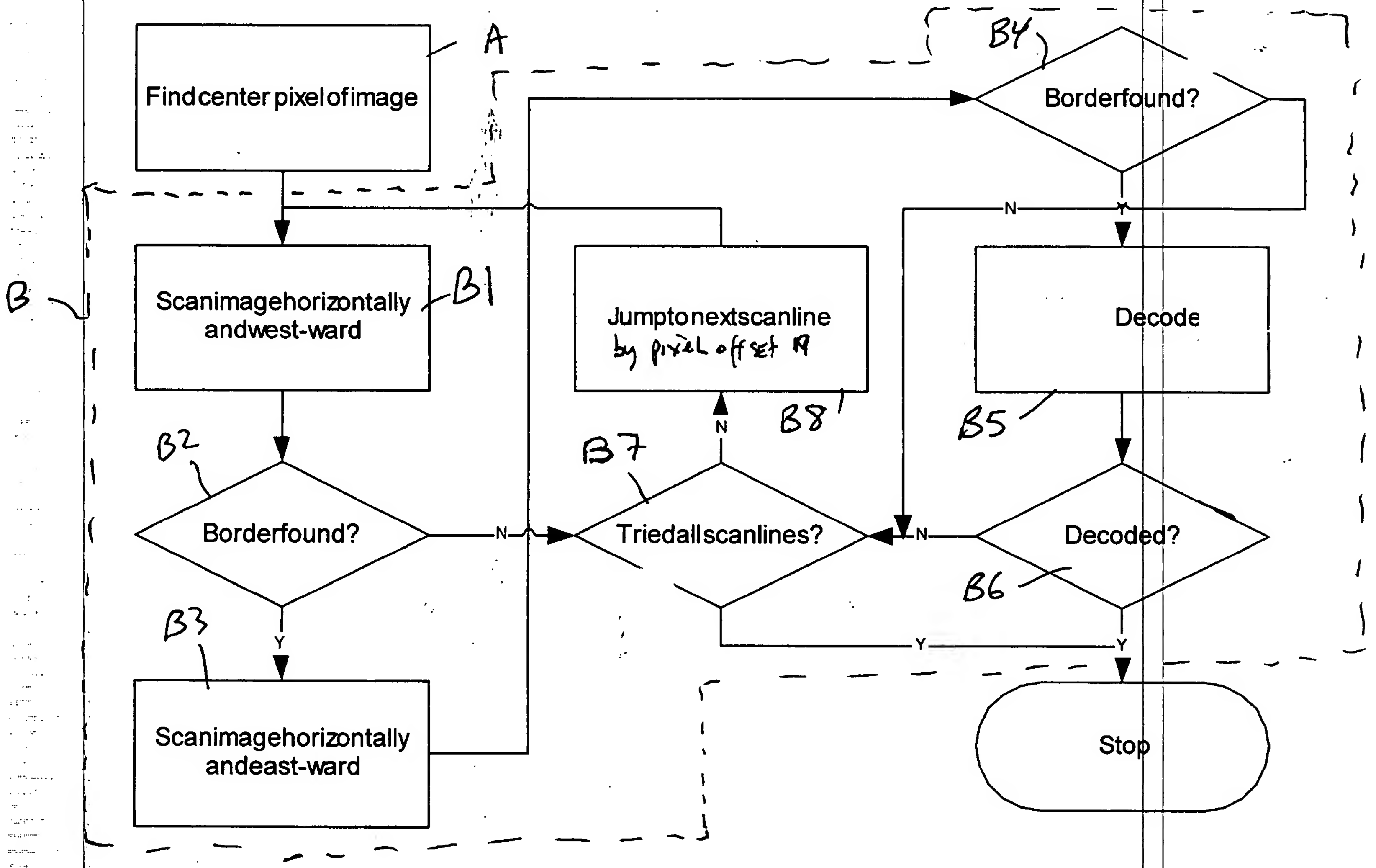
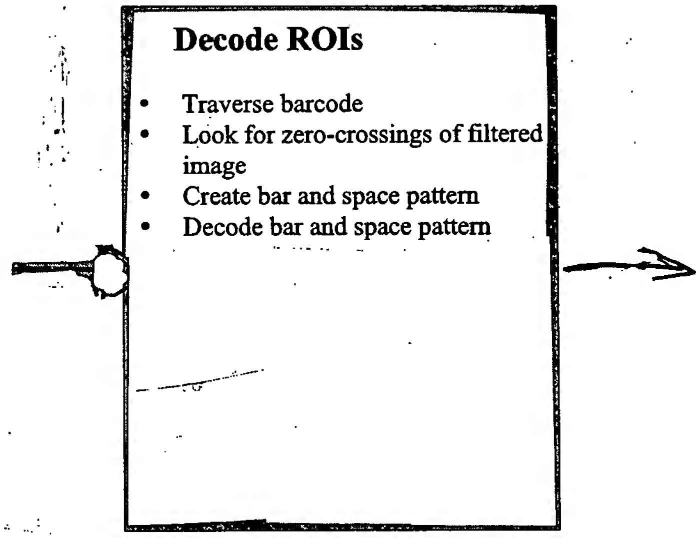


FIG. 20A

Summary of Omniscan mode



Decode ROIs

- Traverse barcode
- Look for zero-crossings of filtered image
- Create bar and space pattern
- Decode bar and space pattern

- No Finder module
- No Marker module
- No Tracker module

- wide area illumination
- assume Barcode is at center!
- 1″ tall
- 1″ wide (aspect ratio = 1)
- 1D

6 angles (50 pixel spacing)
- 0°
- 30°
- 60°
- 90°
- 120°
- 150°

FIG. 21A

OmniScan mode flow-chart

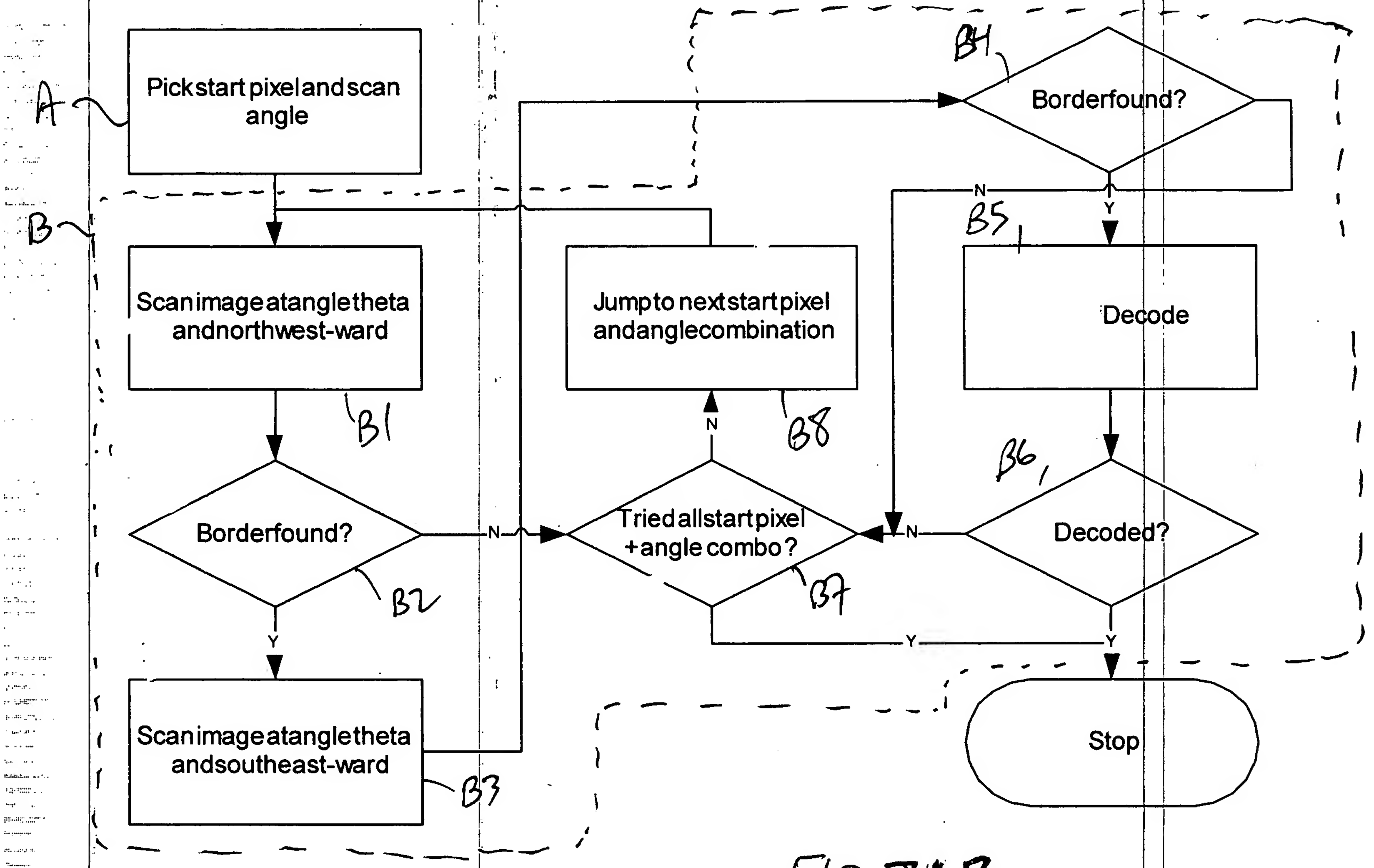
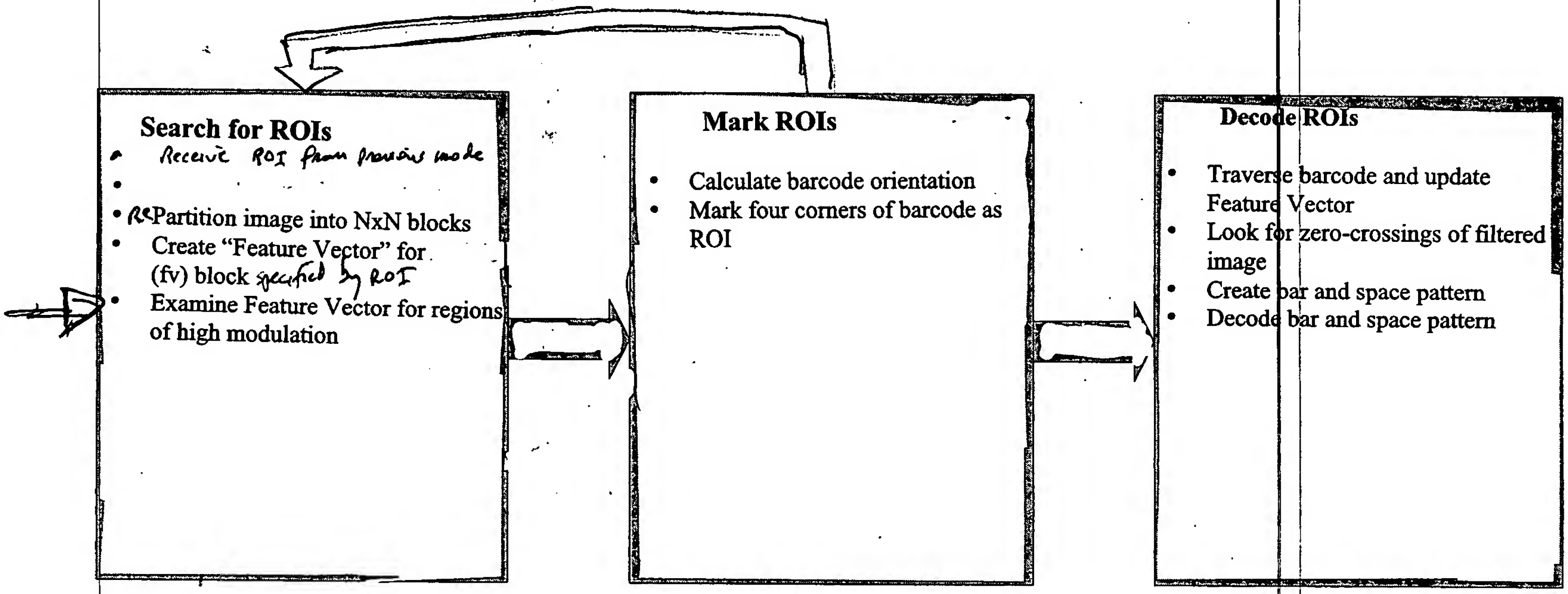

A
Pickstart pixelandscan angle
B
Scanimageatangletheta andnorthwest-ward
B1
Borderfound?
B2
N
Y
Scanimageatangletheta andsoutheast-ward
B3
Jumpto nextstartpixel andanglecombination
B8
N
Triedallstartpixel +angle combo?
B7
Y
B4
Borderfound?
N
Y
B5
Decode
B6
Decoded?
N
Y
Stop

FIG. 21B

Summary of ROI-SPECIFIC mode



Search for ROIs

- Receive ROI from previous mode
-
- RePartition image into NxN blocks
- Create "Feature Vector" for (fv) block specified by ROI
- Examine Feature Vector for regions of high modulation

Mark ROIs

- Calculate barcode orientation
- Mark four corners of barcode as ROI

Decode ROIs

- Traverse barcode and update Feature Vector
- Look for zero-crossings of filtered image
- Create bar and space pattern
- Decode bar and space pattern

FIG. 22A

ROI-SPECIFIC mode flow-chart

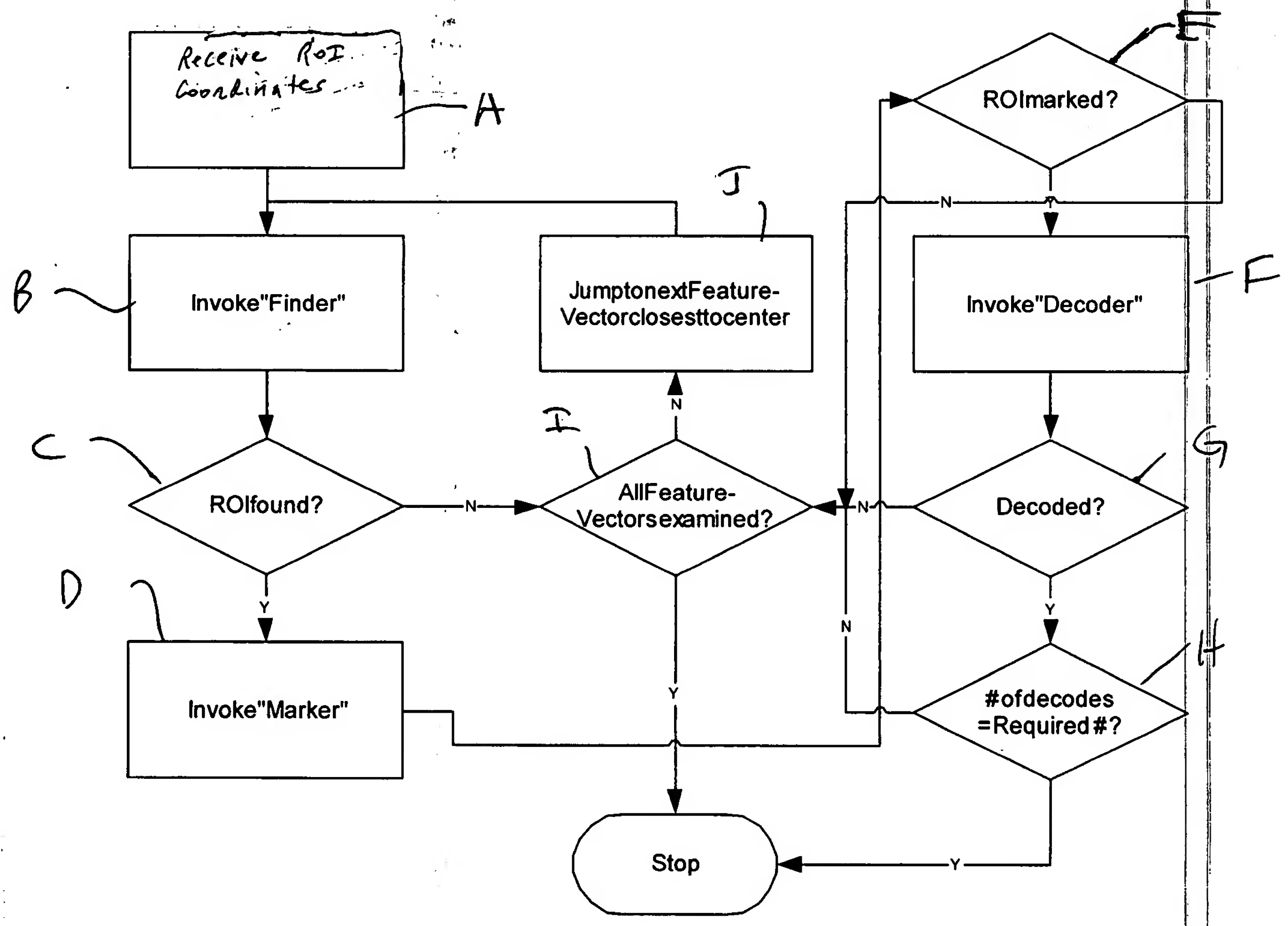


FIG. 22B

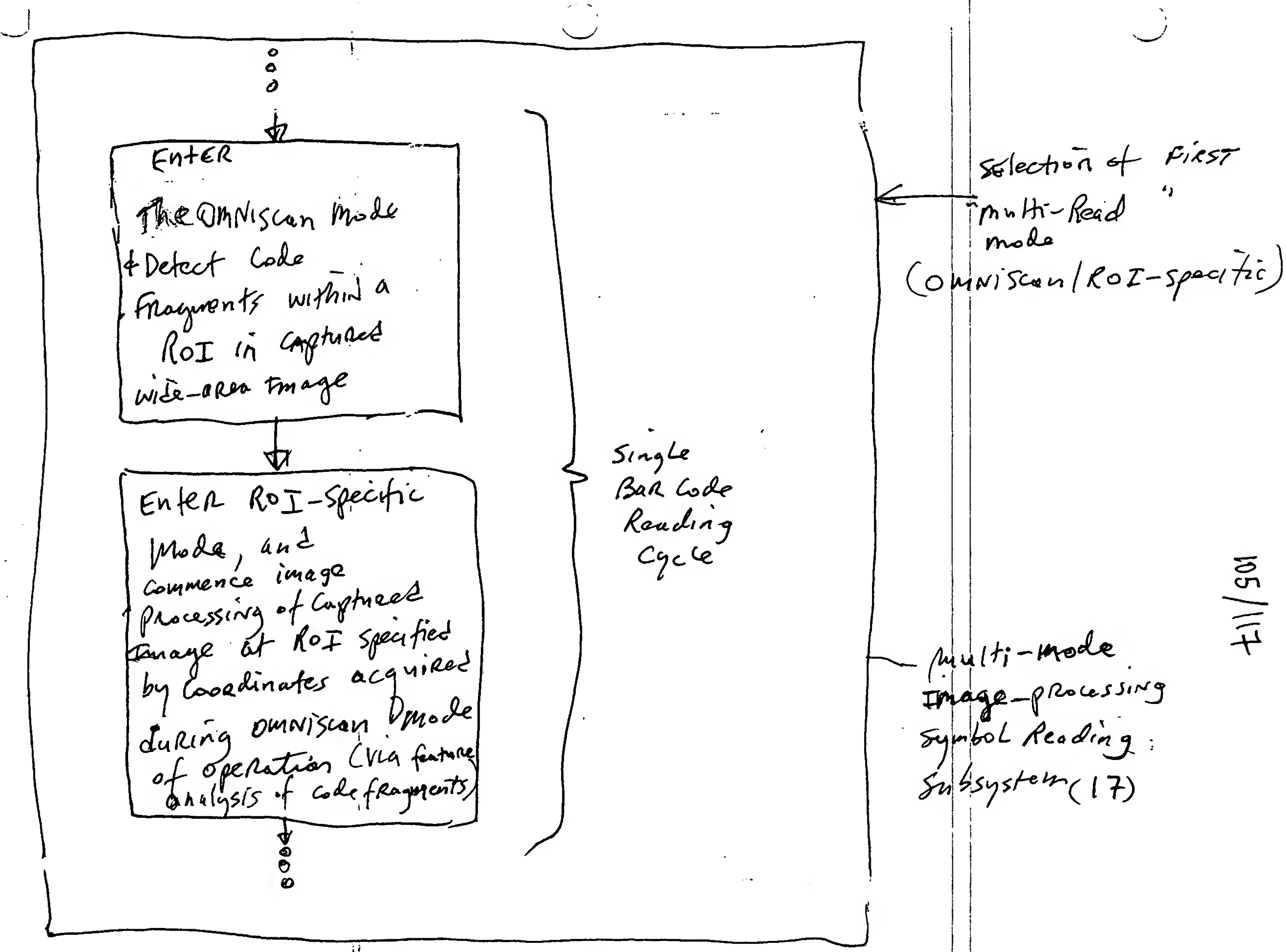

ENTER
The OMNIScan Mode & Detect Code Fragments within a RoI in Captured wide-area Image
Enter RoI-Specific Mode, and commence image processing of Captured Image at RoI specified by Coordinates acquired during OMNIScan mode of operation (via feature analysis of code fragments)
Single Bar Code Reading Cycle
Selection of First "multi-Read" mode (OMNIScan/RoI-specific)
Multi-mode Image-processing Symbol Reading Subsystem (17)

FIG 23

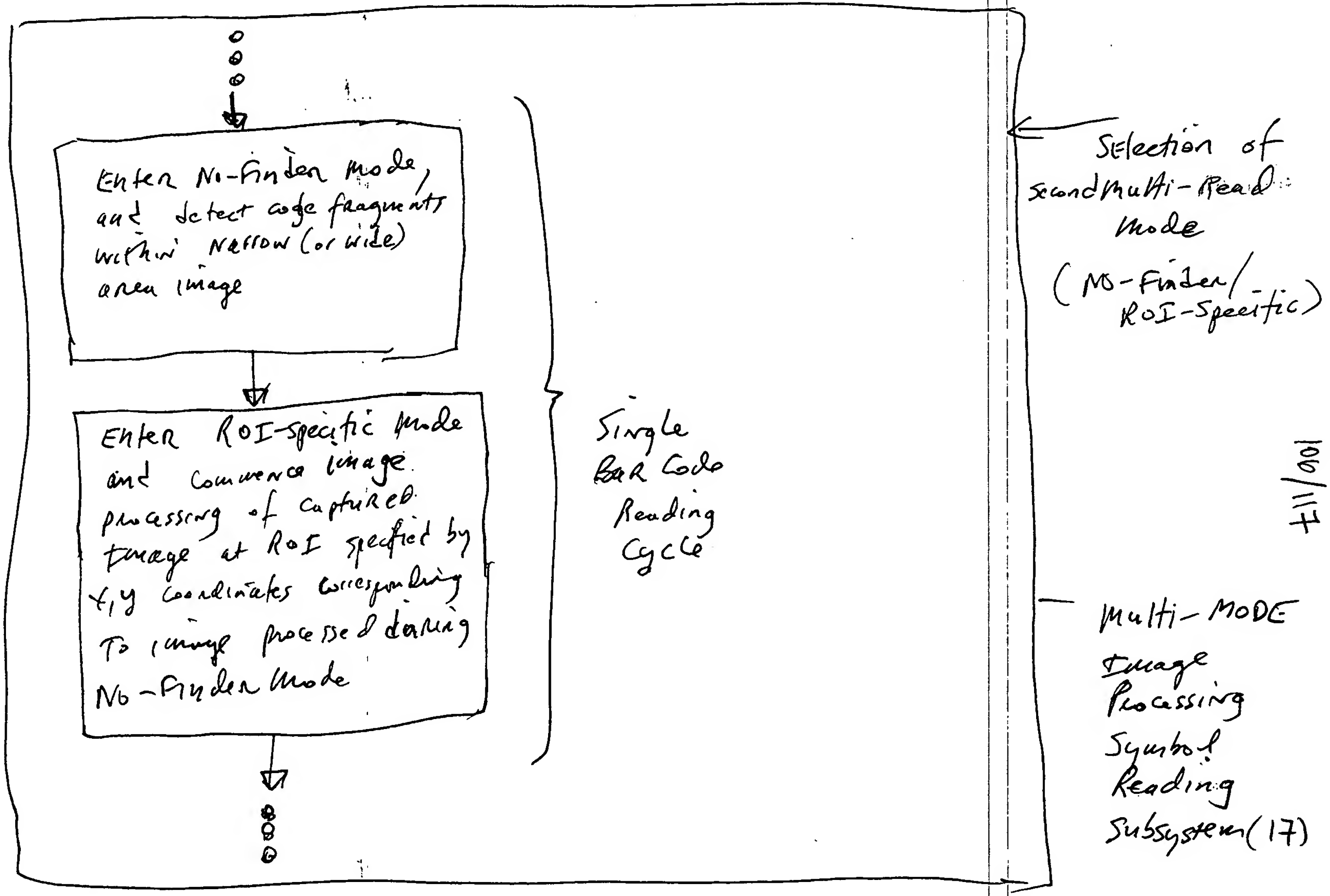

Enter No-Finder mode, and detect code fragments within narrow (or wide) area image
Enter ROI-specific mode and commence image processing of captured image at ROI specified by x,y coordinates corresponding to image processed during No-Finder mode
Single Bar Code Reading Cycle
Selection of second multi-Read mode (No-Finder/ROI-Specific)
Multi-MODE Image Processing Symbol Reading Subsystem (17)

FIG. 24

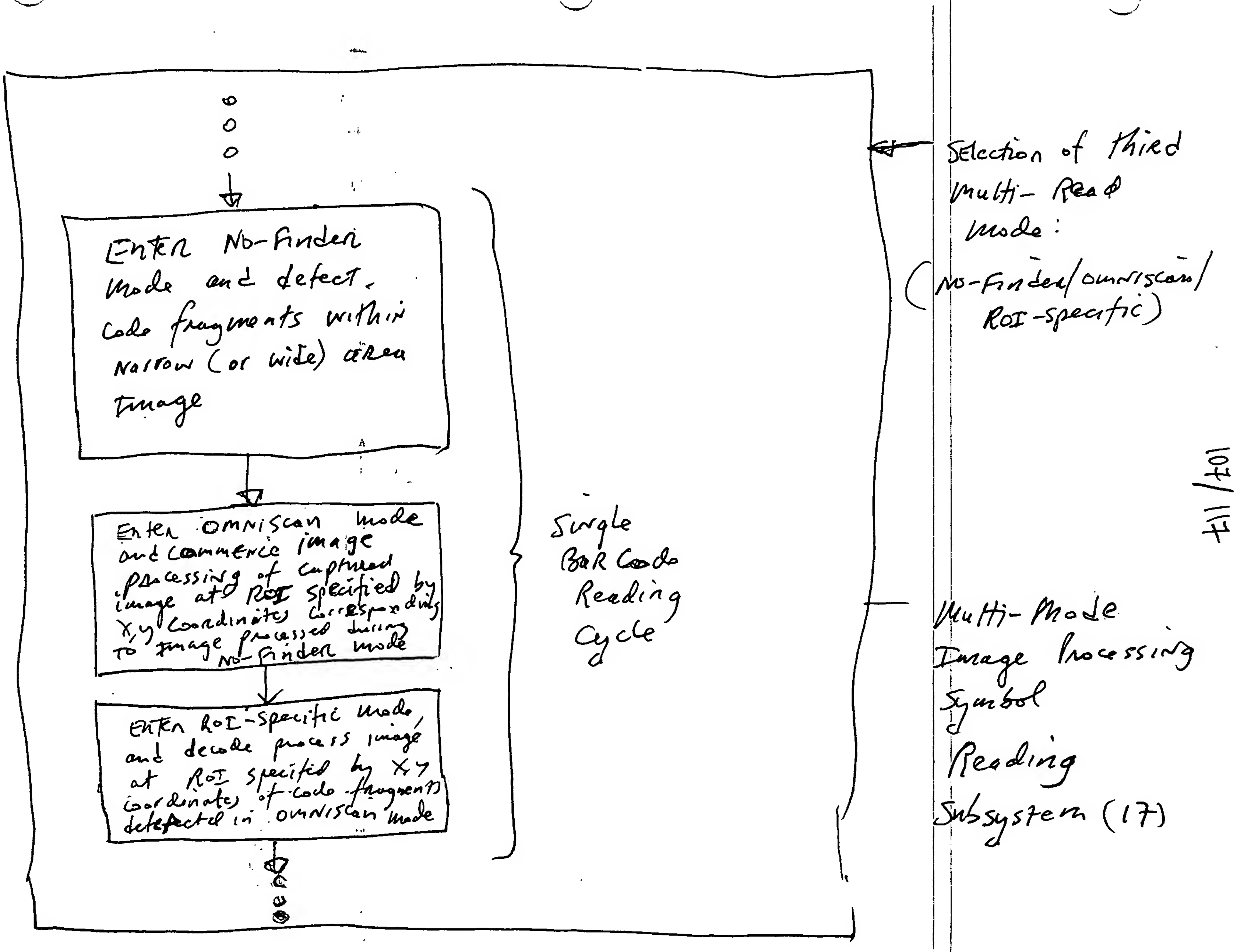

Selection of third
Multi- Read
mode:
(No-Finder/omniscan/
ROI-specific)
ENTER No-Finder mode and defect. code fragments within Narrow (or wide) area Image
Enter omniscan mode and commence image processing of captured image at ROI specified by X,Y coordinates corresponding to image processed during No-Finder mode
Enter ROI-specific mode, and decode process image at ROI specified by X,Y coordinates of code fragments detected in omniscan mode
Single
Bar Code
Reading
Cycle
Multi-Mode
Image Processing
Symbol
Reading
Subsystem (17)

FIG. 25

PROGRAMMABLE MODES OF BAR CODE SYMBOL READING OPERATION WITHIN THE HAND-SUPPORTABLE DIGITAL IMAGING-BASED BAR CODE SYMBOL READER OF THE PRESENT INVENTION

Programmed Mode of System Operation No. 1: Manually-Triggered Single-Attempt 1D Single-Read Mode Employing the No-Finder Mode of Operation

~~**Programmed Mode Of System Operation No. 2: Manually-Triggered Multiple-Attempt 1D Single-Read Mode Employing the No-Finder Mode of Operation**~~

Programmed Mode Of System Operation No. 3: Manually-Triggered Single-Attempt 1D/2D Single-Read Mode Employing the No-Finder And The Automatic Or Manual Modes of Operation

Programmed Mode of System Operation No. 4: Manually-Triggered Multiple-Attempt 1D/2D Single-Read Mode Employing the No-Finder And The Automatic Or Manual Modes of Operation

Programmed Mode of System Operation No. 5: Manually-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing the No-Finder And The Automatic Or Manual Modes of Operation

Programmed Mode of System Operation No. 6: Automatically-Triggered Single-Attempt 1D Single-Read Mode Employing The No-Finder Mode Of Operation

Programmed Mode of System Operation No. 7: Automatically-Triggered Multi-Attempt 1D Single-Read Mode Employing The No-Finder Mode Of Operation

Programmed Mode of System Operation No. 8: Automatically-Triggered Multi-Attempt 1D/2D Single-Read Mode Employing The No-Finder and Manual and/or Automatic Modes Of Operation

Programmed Mode of System Operation No. 9: Automatically-Triggered Multi-Attempt 1D/2D Multiple-Read Mode Employing The No-Finder and Manual and/or Automatic Modes Of Operation

Programmable Mode of System Operation No. 10: Automatically-Triggered Multiple-Attempt 1D/2D Single-Read Mode Employing The Manual, Automatic or Omniscan Modes Of Operation

Programmed Mode of System Operation No. 11: Semi-Automatic-Triggered Single-Attempt 1D/2D Single-Read Mode Employing The No-Finder And The Automatic Or Manual Modes Of Operation

FIG. 26A

Programmable Mode of System Operation No. 12: Semi-Automatic-Triggered Multiple-Attempt 1D/2D Single-Read Mode Employing The No-Finder And The Automatic Or Manual Modes Of Operation

Semi-Automatic-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing The No-Finder And The Automatic Or Manual Modes Of Decoder Operation; Programmable Mode of Operation No. 13

Programmable Mode of Operation No. 14: Semi-Automatic-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing The No-Finder And The Omniscan Modes Of Operation

Programmable Mode of Operation No. 15: Continously-Automatically-Triggered Multiple-Attempt 1D/2D Multiple-Read Mode Employing The Automatic, Manual Or Omniscan Modes Of Operation

Programmable Mode of System Operation No. 16: Diagnostic Mode Of Imaging-Based Bar Code Reader Operation

Programmable Mode of System Operation No. 17: Live Video Mode Of Imaging-Based Bar Code Reader Operation

FIG. 26B

Imaging-Based Bar Code Symbol Reading System with Extended Multi-mode Illumination Subsystem

- Four modes of Illumination:

(1) Wide-Area for "near" object (0 mm – 100 mm)
(2) Wide-Area for "far" object (100 mm – 200 mm)
(3) Narrow-Area for near object (0 mm – 100 mm)
(4) Narrow-Area for far object (100 mm – 200 mm)

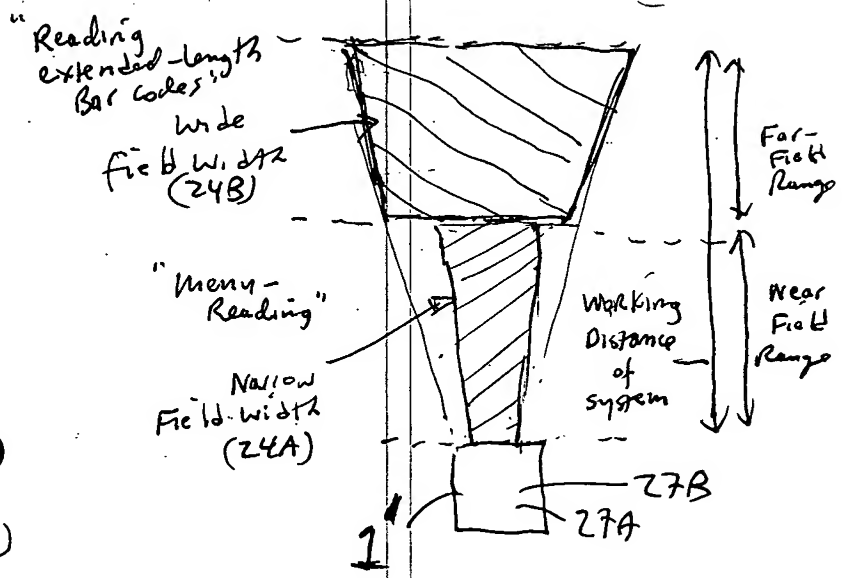


FIG. 27B

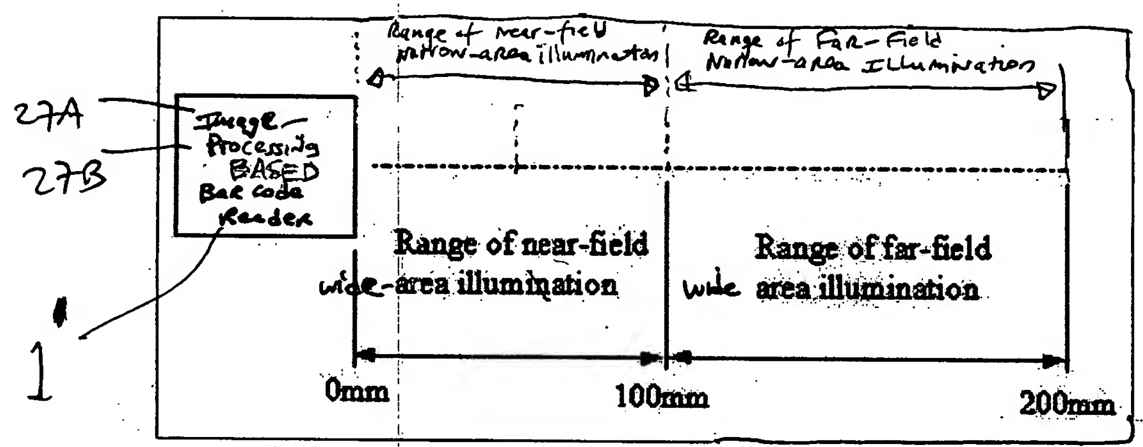

Range of near-field narrow-area illumination
Range of Far-Field Narrow-area Illumination
Image-Processing BASED Bar code Reader
27A
27B
1
Range of near-field wide-area illumination
Range of far-field wide area illumination
0mm
100mm
200mm

FIG. 27A

LED Arrangements For NEAR-Field and FAR-Field WIDE AREA ILLUMINATION ARRAYS and Narrow-AREA Illumination ArrayS

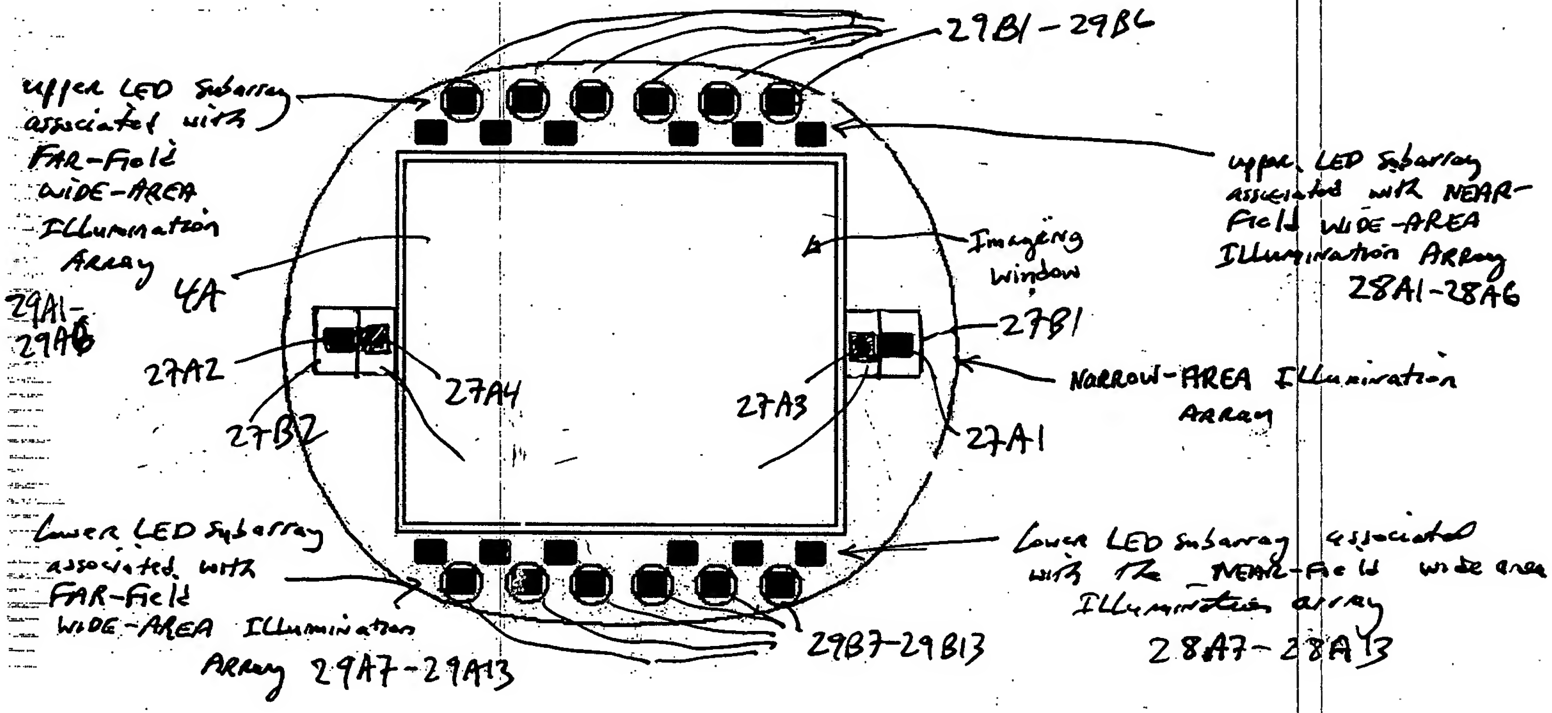

29B1-29B6
upper LED Subarray associated with FAR-Field WIDE-AREA Illumination Array
29A1-29A6
4A
upper LED Subarray associated with NEAR-Field WIDE-AREA Illumination Array 28A1-28A6
Imaging window
27B1
NARROW-AREA Illumination Array
27A2
27A4
27B2
27A3
27A1
Lower LED Subarray associated with FAR-Field WIDE-AREA Illumination Array 29A7-29A13
29B7-29B13
Lower LED Subarray associated with the NEAR-Field wide area Illumination array 28A7-28A13

FIG. 28

Hands-Free Mode of operation

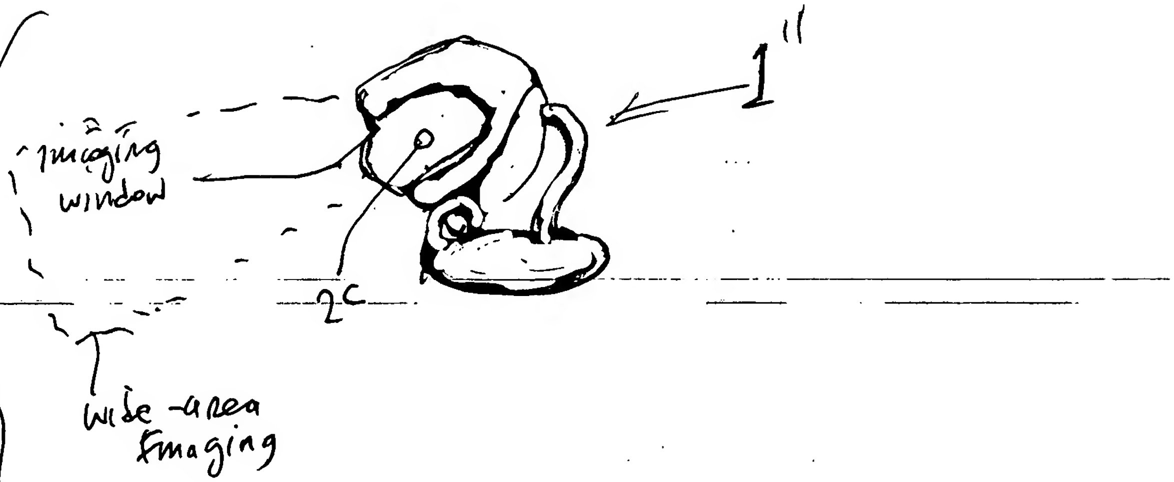


FIG. 29A

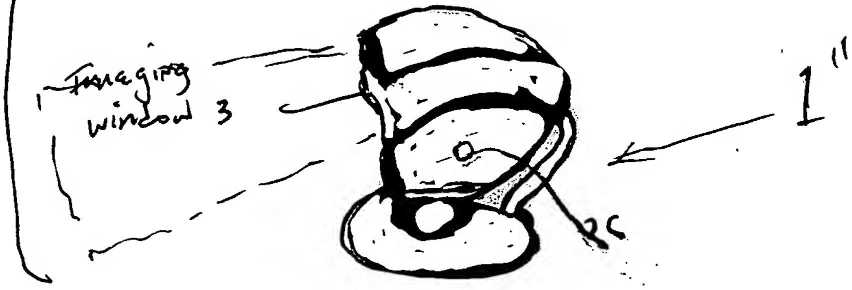


FIG. 29B

Hands-ON Mode of operation

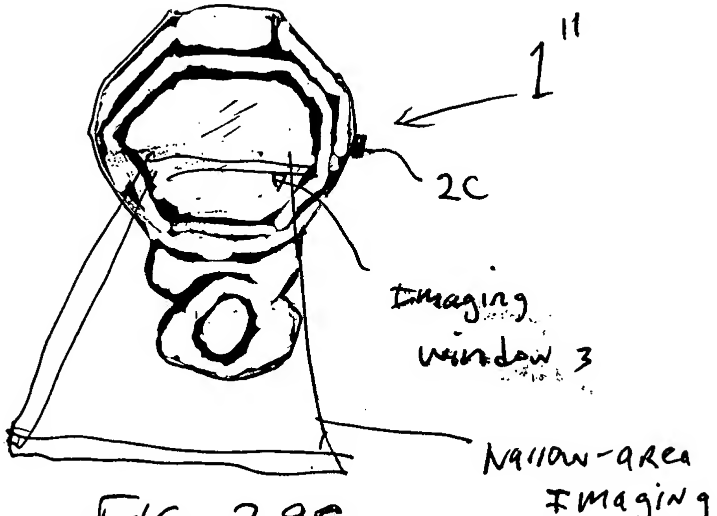


FIG. 29C

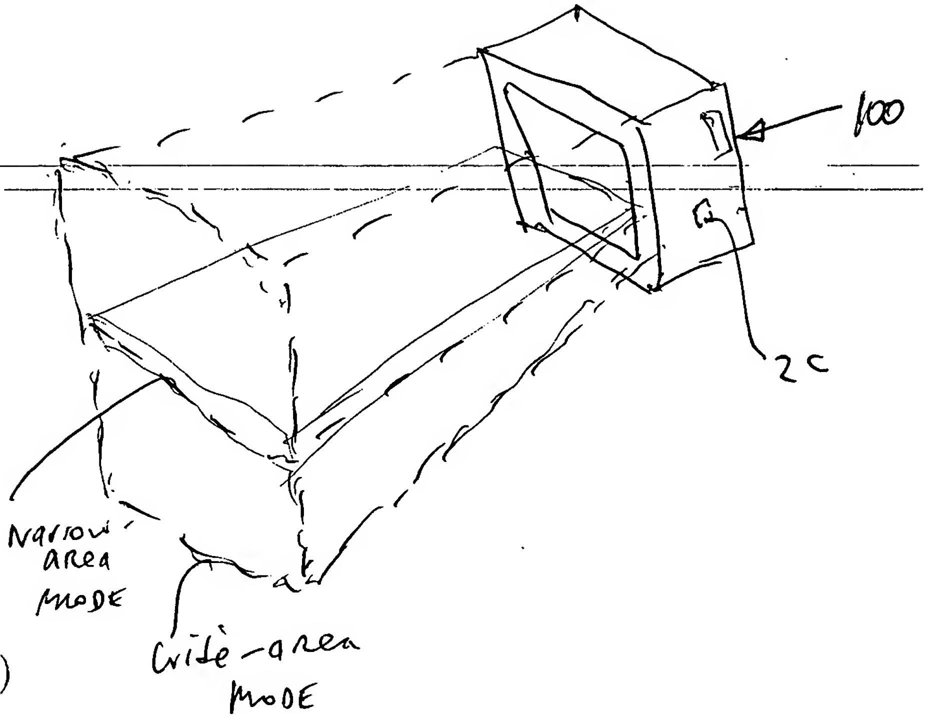
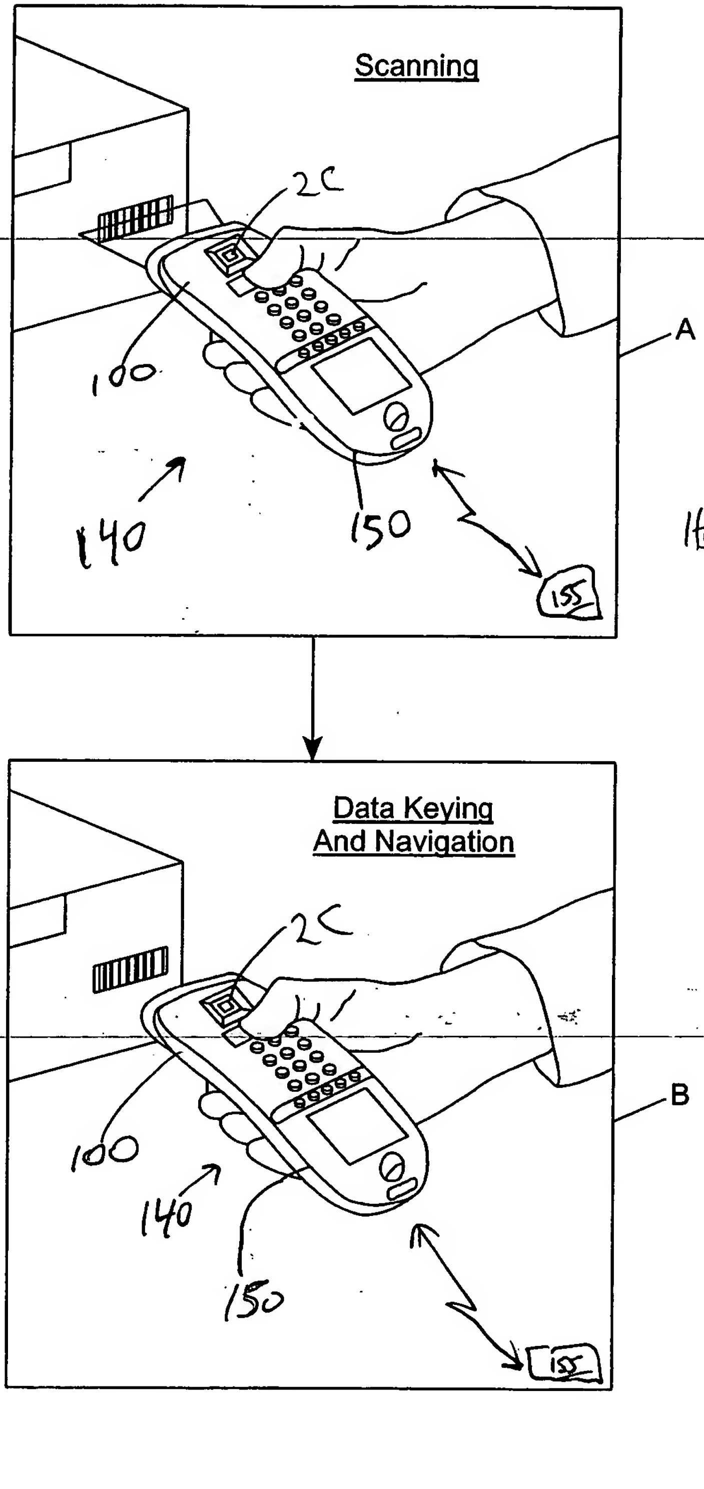


FIG. 30



Hands-on mode of operation

FIG 31

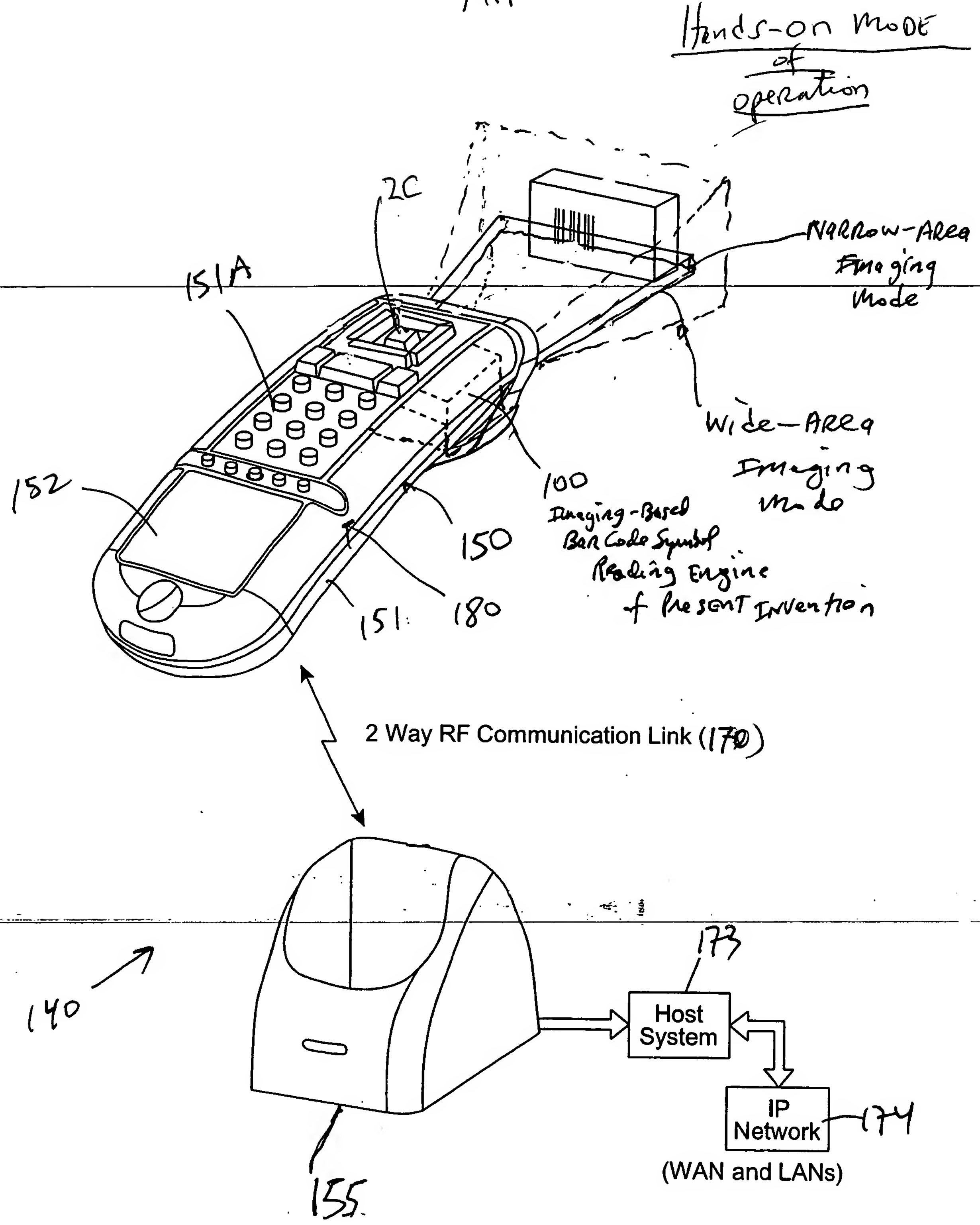
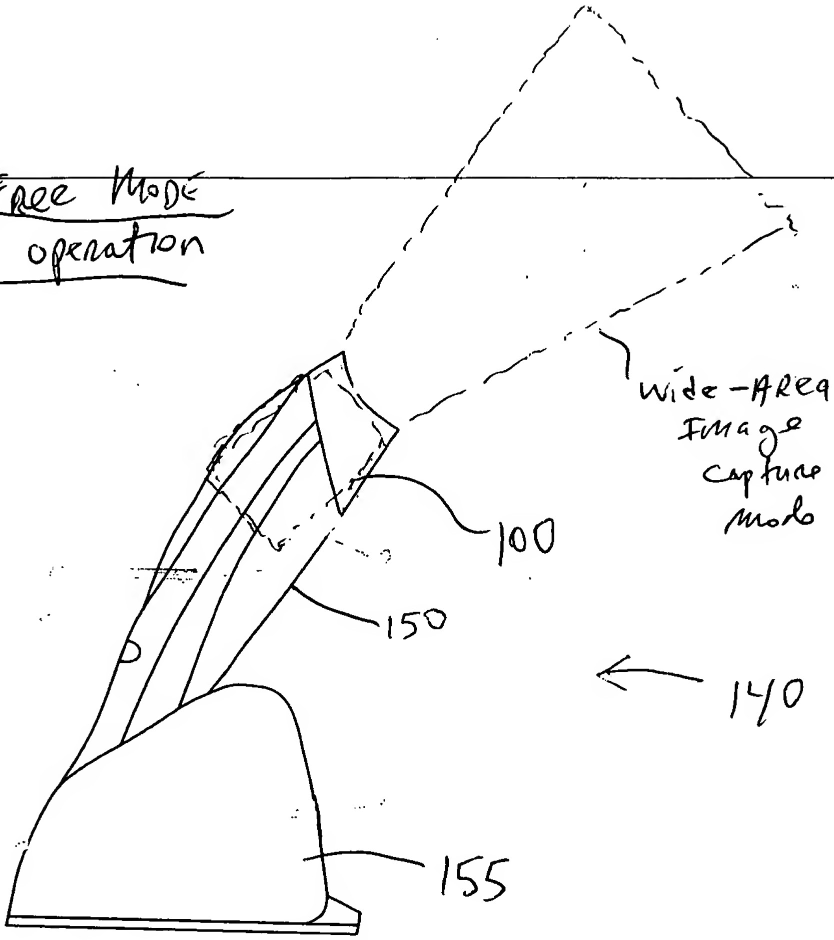

Hands-on Mode of operation
2C
151A
Narrow-Area Imaging Mode
Wide-Area Imaging Mode
152
100
Imaging-Based Bar Code Symbol Reading Engine of Present Invention
150
151
180
2 Way RF Communication Link (170)
140
173
Host System
IP Network
174
(WAN and LANs)
155

FIG. 32



Hands-Free Mode of operation

FIG. 33

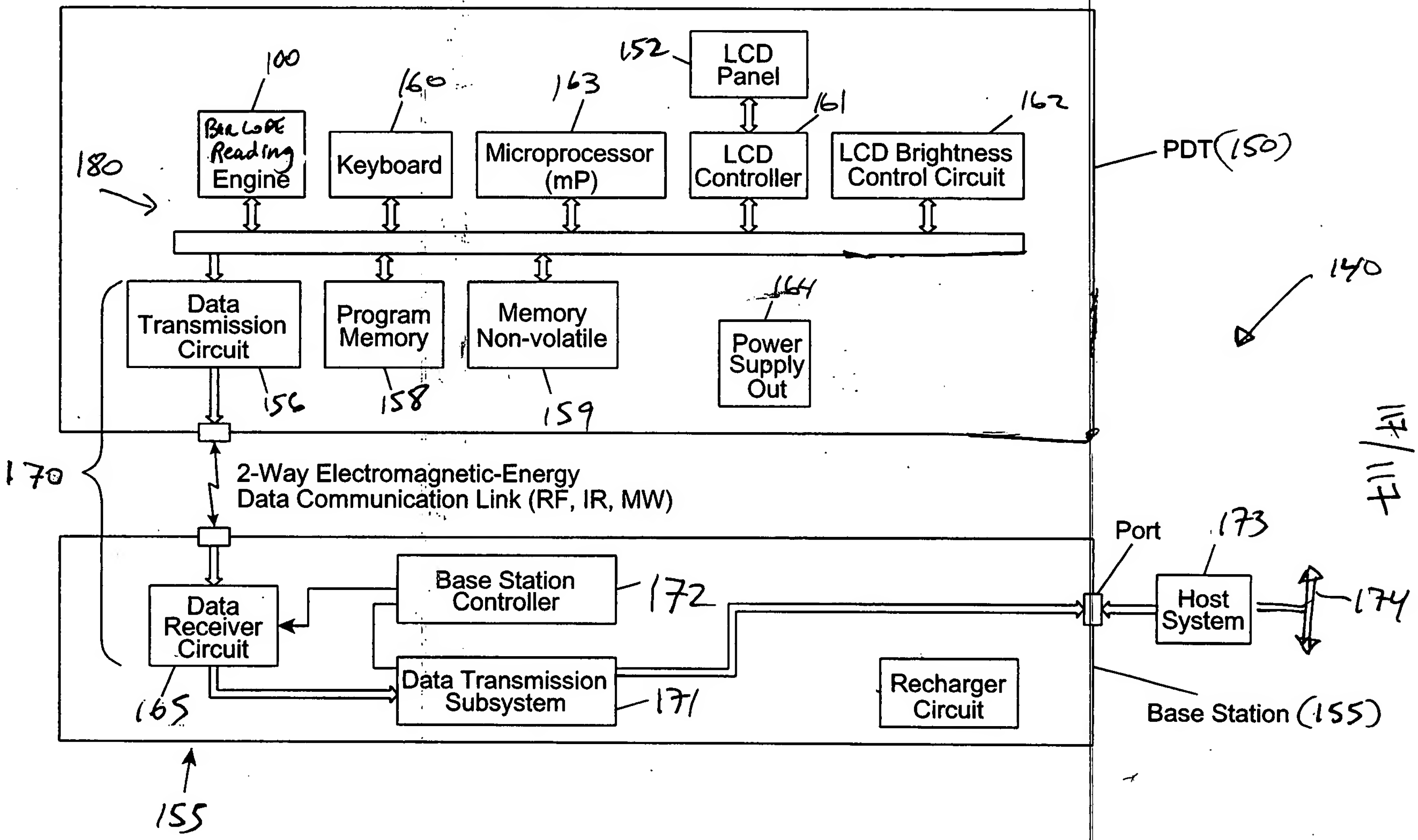

100
160
163
152
161
162
180
Bar Code Reading Engine
Keyboard
Microprocessor (mP)
LCD Panel
LCD Controller
LCD Brightness Control Circuit
PDT (150)
140
164
Data Transmission Circuit
Program Memory
Memory Non-volatile
Power Supply Out
156
158
159
170
2-Way Electromagnetic-Energy Data Communication Link (RF, IR, MW)
Port
173
Base Station Controller
172
Host System
174
Data Receiver Circuit
Data Transmission Subsystem
171
Recharger Circuit
165
Base Station (155)
155

FIG 33